ADVANCES IN DISCRETE GEOMETRY APPLIED TO THE EXTRACTION OF PLANES AND SURFACES FROM 3D VOLUMES

THÈSE N^{O} 1944 (1999)

PRÉSENTÉE AU DÉPARTEMENT D'INFORMATIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES TECHNIQUES

PAR

Oscar FIGUEIREDO

Ingénieur physicien-électronicien ICPI/CPE Lyon, France
DEA d'informatique, Université de St Etienne, France
de nationalités française et portugaise

acceptée sur proposition du jury:

Prof. Roger-D. Hersch, rapporteur
Prof. Jean-Pierre Reveillès, corapporteur
Dr. Daniel Cohen-Or, corapporteur
Dr. Jean-Philippe Thiran, corapporteur

Lausanne, EPFL
1999

A mes parents

“Geometry is not there for being learned. It is there for being used.”

Seymour Papert, *The Children’s Machine*

Table of Contents

Medical Image Visualization Applications

Acknowledgements

First and foremost, I am particularly grateful to Professor Roger David Hersch for trusting me and giving me the opportunity to complete this work at the Peripheral Systems Laboratory (LSP) under his direction. His continuous support and guidance were an incomparable stimulation and helped me to reach the objectives and to avoid the pitfalls along the way.

Professor Jean-Pierre Reveillès initiated me into the art of discrete geometry. Many times I thought to myself how lucky I was to have met him, not only because I consider his approach to discrete geometry the most interesting and the most promising but also because of his qualities as a man. His presence despite the distance, his careful advice and inspiration, his ever renewed enthusiasm made it a pleasure to work with him.

I would like to give special thanks to Dr. Benoît Gennart for all the enlightening discussions I had with him. I can not remember having left his office once without at least a new direction to tackle a problem I submitted to him.

Dr. Vincent Messerli and Dr. Benoît Gennart are the main authors of the CAP/PS^2 parallelization framework. Without their own work, the present one could not have found its full expression.

All the project partners, especially Professor O. Ratibe, Dr. R. Welz, Dr. L. Bidaut and A. Farine were of great help in the design of the medical applications.

I would like to thank also Marc Mazzariol, Olivier Courtois, Dr. Patrick Emmel, Dr. Changyuan Hu, Joaquin Tarraga and all the LSP staff for making the LSP such a nice place to work in, and not only...

All my family and friends, those who are in Portugal, those who are in France and those I met in Switzerland, all of them helped me along the years, sometimes without knowing how much.

Finally I thank the Swiss National Fund and the E.P.F.L. for funding this research.

Summary

A wide range of human activities relies on 2D and 3D image processing, synthesis and display. Medicine is for instance one of the fields where imaging has gained particular importance in the recent past because of the strong demand for non-invasive exploration techniques primarily based on tomographic imaging.

With the advent of computers and digital technologies, the vast majority of the images manipulated today in medicine as well as in other disciplines is composed of discrete samples, called *pixels* (or *voxels* in the case of volumic images). In order to exploit these images on computers efficiently, several challenges must be met:

- traditional euclidean geometry, based on continuous concepts, appears not to be well-suited to computers and images which are intrisically discrete. Geometric operations based on floating-point arithmetic often lead to inconsistent results and algorithmic instability
- the increasing size of the images, especially three-dimensional, creates an I/O bandwidth problem: images that do not entirely fit in memory must be read from comparatively slow mass storage devices

In this context, a solid theoretical basis is needed to solve the problems introduced by the inadequacy of traditional euclidean geometry with respect to digital images. Discrete geometry is a young and active branch of mathematics that aims at building such a theoretical foundation for a consistent description of digital objects and operations. By nature, the results it establishes are particularly well-suited to computer processing and lead to simple and efficient integer-based algorithms that can be parallelized including at the I/O level, thus bringing increased performance and a solution to the I/O bottleneck problem.

In the first half of this work, we propose new theoretical results and definitions for objects like three-dimensional digital lines, digital spline curves and surface patches as well as algorithms for rigorously solving problems like the intersection of 3D digital lines and planes or the determination of the covering of digital lines and parallelograms by rectangular plane tesselations.

In the second half, we emphasize the strengths of this approach by introducing two concrete applications of these results in the field of medical imaging, namely the extraction of planes of arbitrary orientation and of ruled surfaces from very large 3D discrete volumes (several Gigabytes). These algorithms derived from discrete geometry are implemented on parallel architectures consisting of commodity components (standard PCs with multiple SCSI disks connected through Fast Ethernet). They achieve remarkable performance and scalability. Thus,

on a configuration consisting of a master and five slave nodes (Dual PentiumPro at 200MHz with 12 disks each), the plane extraction algorithm is able to extract close to 5 slices per second (512x512 pixels, 24 bits color). On a more modest configuration consisting of a Dual-Pentium II at 300MHz with 16 disks, the same algorithm has also proven its stability and performance by serving Internet users and performing approximately 160’000 plane extraction requests from the Visible Human body (13 GB) in 6 months (see `http://visiblehuman.epfl.ch/`).

Résumé

Un grand nombre d'activités humaines reposent aujourd'hui sur le traitement, la synthèse et l'affichage d'images. La médecine est ainsi un des domaines où l'imagerie a gagné une importance particulière ces dernières années à cause de la forte demande en techniques non invasives de diagnostic et de traitement, reposant essentiellement sur l'imagerie tomographique.

Avec l'avènement de l'informatique et des technologies numériques, l'immense majorité des images ainsi manipulées aujourd'hui en médecine comme dans d'autres domaines est composée de points discrets appelés *pixels* (ou *voxels* dans le cas d'images volumiques). L'exploitation informatique de telles images doit, pour être efficace, faire face à plusieurs défis:

- la géométrie euclidienne classique, basée sur des concepts continus, se révèle peu adaptée à des images et des traitements par ordinateur qui sont intrinsèquement discrets. Les opérations géometriques basées sur l'arithmétique en nombre flottants apparaissent ainsi plus délicates et conduisent souvent à des résultats incohérents ou des instabilités algorithmiques.
- l'accroissement de la taille des images à traiter, en particulier 3D, pose un problème d'entrées/sorties lorsque les images deviennent trop grandes pour être entièrement contenues en mémoire et doivent être lues depuis des périphériques de stockage comparativement plus lents.

Dans ce contexte, une base théorique solide permettant de résoudre les problèmes liés à l'inadaptation de la géométrie euclidienne classique aux images numériques devient nécessaire. La géométrie discrète est une branche émergente et très active des mathématiques dont le but est de construire une telle fondation pour la description des objets et opérations discrets. Par nature les résultats qui en dérivent sont particulièrement adaptés à une implémentation informatique et conduisent à des algorithmes en nombres entiers qui peuvent être plus facilement parallélisés y compris au niveau de l'accès aux données, apportant ainsi une solution au problème de congestion des entrées/sorties.

Dans la première moitié de cet ouvrage, nous proposons donc un certain nombre de nouveaux résultats et définitions théoriques concernant des objets comme les droites discrètes 3D, les courbes et surfaces splines discrètes ainsi que des algorithmes permettant de résoudre de façon rigoureuse des problèmes tels que l'intersection de droites 3D et de plans discrets ou encore la détermination de la couverture de droites et parallélogrammes discrets par des pavages rectangulaires du plan.

Dans la seconde moitié, nous mettons en évidence le potentiel de cette approche à travers deux applications concrètes des résultats précédents dans le domaine de l'imagerie médicale, à savoir l'extraction de plans de coupe d'orientation quelconque et de surfaces réglées de l'intérieur de volumes discrets de très grandes dimensions (plusieurs Gigaoctets). Ces algorithmes dérivés de la géométrie discrète et implémentés sur des architectures parallèles à base de composants grand public (PCs munis de grappes de disques SCSI et connectés par Fast Ethernet) atteignent des performances remarquables et tirent au mieux partie de la puissance du matériel disponible allant du simple PC isolé à la configuration haut de gamme constituée d'un réseau local de machines. Ainsi, sur une configuration composée d'un maitre et de cinq esclaves (bi-Pentium Pro à 200MHz avec douze disques chacun), l'algorithme d'extraction de plan est capable d'extraire près de cinq images par seconde (512x512 pixels, 24 bits couleur). Sur une configuration plus modeste composée d'un unique bi-Pentium II à 300 Mhz avec 16 disques, le même algorithme a prouvé sa grande stabilité et sa performance en servant, en six mois, aux Internautes de par le monde, plus 160 000 requêtes d'extraction de plan de l'intérieur du Visible Human (13 Go) (voir le serveur "*Le corps humain sous tous ses angles*": `http://visiblehuman.epfl.ch/`)

Preface

Down to its theoretical roots in Boolean algebra and set theory, computer science is probably the discipline that has accepted the fewer compromises with the continuum. In a surrounding universe which constantly hesitates between the discrete steps of quantum physics and the continuous moves of Newton's and Einstein's mechanics, computers and their silicon chip cores beating with their invariable 0s and 1s have founded the apparently immovable reign of digital information and computation. And all the physical phenomena that not so long ago were granted as continuous have now become discrete, victims of sampling and quantization. Examples are innumerable: digital compact discs have eradicated the analog signals recorded on vinyls, digital cameras will probably seal the same fate on traditional photography, radio and television signals will soon follow. From the most elementary to the most complex, every quantity that at some point goes through a computer has little choice but to be sampled and quantized.

At the dawn of this digital revolution, new disciplines like information theory were born while other topics in mathematics and signal theory that would have otherwise been neglected, have gained new importance. Among these, digital geometry has raised particular interest among mathematicians and scientists working in computer graphics in the last two decades. This is not surprising since image visualization and processing are nowadays omnipresent in almost all human activities. From entertainment (computer games, movie pictures) to scientific disciplines (medical imaging, satellite imaging, astronomy) and industrial activities (computer aided design, quality control), saying that digital images are everywhere has become a commonplace. This overwhelming presence of the image drives strong demands for algorithms both for image synthesis and analysis. These two main algorithmic classes have a common denominator: geometry. Images are indeed often synthesized by means of geometric primitives such as straight line segments, curved lines, surfaces and geometric operations such as scalings, rotations, shearings, etc... On the other hand, image analysis often requires identifying those same primitive geometric objects in order to extract valuable information.

Every pupil is taught about euclidean geometry right from his/her early years at school and knows about lines, triangles, circles and their geometric properties: intersections, symmetries, etc... Later on, students learn about algebraization of these geometric notions and know how to represent these objects and their properties by means of equations and numbers. So quite naturally, people tend to have high hopes of getting good results when applying these familiar and so carefully learned notions to computer images. Unfortunately this approach is often disappointing and the challenges posed by discrete geometrical structures soon appear to be considerable.

The simplest questions of euclidean geometry become apparently inextricable when considered in a discrete space: the very definition of a straight line as the shortest path between two points does not hold anymore, digital straight lines may have more than one intersection point (even worse: the set of points making up this intersection may be disconnected), in a discrete space, a simple operation like a rotation is anything but easy to define since ensuring properties like shape preservation is really arduous. For many, these difficulties arise from the fact that discrete geometry is considered as an approximation or degenerate case of the more familiar euclidean geometry. Recent experience has shown on the contrary, that there is most to gain in building a fresh new theory aimed at studying the intrisic properties of digital structures independently of euclidean geometry approximations.

Discrete geometry is this branch of mathematics that has developed and grown out of the difficulties and frustrations engendered by the manipulation of continuous concepts on discrete computing machines. Initially driven by practical considerations exclusively, discrete geometry has now become a fully-fledged mathematical theory of its own but it still finds its ultimate beauty in concrete applications. And even though it can find lots of other application fields, it is naturally particularly well-suited to computer imaging problems.

Discrete geometry is a discipline in its infancy when compared to the twenty-three centuries of euclidean geometry. It is a vast open research field where the questions are many and the answers only a few. Research proceeds in a wide variety of areas and the synthesis effort that characterizes mature sciences will probably come in a few years only. Therefore the first half of this work does not attempt at presenting a consistent set of results focused towards a specific objective but rather a collection of selected topics: intersection of 2D digital lines, definition and properties of 3D digital lines, digitization of Bézier curves and surfaces and finally the covering of discrete structures (digital lines and parallelograms) by rectangular tessellations of the plane.

As we underlined before, developments in discrete geometry are often driven by practical computing needs and this discipline finds its ultimate full expression in concrete applications. Therefore, in the second half of this work, we show how discrete geometry and especially some of the results presented in the first half were used to produce two high-performance medical imaging applications. *DigiPlan* is a library for the extraction of planes of arbitrary orientation out of 3D discrete volumes and *DigiSurf* is a library that generalizes the former and allows to extract ruled surfaces out of 3D discrete volumes.

Though strictly speaking, these algorithms are perfectly suitable for other disciplines (e.g., physics or geology), medicine which is a very demanding domain for computer imaging techniques is deemed a particularly good application field for discrete geometry and has historically driven its early developments. Indeed images produced by medical scanning devices (Computed Tomography, Magnetic Resonance Imaging) are of course discrete and are constantly increasing in resolution and hence in size. This induces new constraints and performance requirements both in terms of processing power and I/O bandwidth. Furthermore, current trends aim at spreading the usage of digital media like CD-ROMs or the Internet as a means of exchange of images between physicians which will require high portability and scalability of visualization applications even on low-end computer systems.

Thanks to the underlying theoretical results in discrete geometry, the proposed algorithms use integers only wherever possible and do not need dedicated hardware. Their simplified and robust algorithmics as well as their integration in the state-of-the-art parallelization software framework CAP/PS^2 made it possible to design parallel implementations that take maximum advantage of the underlying commodity hardware and scale particularly well from the low-end single PC computer up to a network cluster consisting of several PC's with several disks attached.

Discrete Geometry

1 Introduction

In recent years, with the spread of the usage of images generated or processed by computers, a new mathematical theory called *discrete geometry* has emerged beside traditional euclidean geometry. This theory aims at studying objects called *discrete objects* consisting of countable sets of points, whereas euclidean continuous objects generally consist of non-denumerable sets of points. Besides their names, discrete objects have little in common with euclidean geometric objects. Indeed, most elementary results of euclidean geometry do not hold in discrete spaces: fundamental notions like continuity are shaken (what does “continuous” mean in a discrete space ?), even the very definitions of objects become complex (how can a straight line segment be defined in a discrete space ?). Discrete geometry tries to address these problems by devising new results specific to discrete spaces and objects, and transposing the familiar notions of euclidean geometry to this context.

Discrete geometry is more of a set of theories rather than a single theory. If the goals are clear, the ways to get there are many, and the name “discrete geometry” gathers different research directions together: discrete topology, arithmetic geometry, graph theory, theory of cellular automata. Among these tracks, the first two currently seem the most promising and benefit from the work of most researchers in the field.

Discrete topology considers discrete geometrical objects under the topological angle either by means of connectivity relations (see Section 2.1.1 for a definition) or as combinatorial structures [24][53]. On the other hand *arithmetic geometry* tries to link the properties of discrete geometric objects to those of integer numbers. Thus it tries to inherit from one of the oldest theories in mathematics, also one of the most puzzling that apparently manipulates the simplest and the most fundamental concepts, integer numbers, and yet hides perhaps the most arduous problems. Recently, fascinating results about digital lines and planes, quasi-affine transformations, discrete rotations and linearization of geometric objects have generated a growing interest for arithmetic geometry. The approach still promises a wealth of new results and elegant solutions and is seen by some specialists as the future of discrete geometry [24]. Moreover, unlike discrete topology which may be sometimes seen as more abstract and whose connection to concrete applications is not always obvious, arithmetic geometry, closely tied to arithmetic calculus, is often more algorithmic and more directly linked to computer implementation.

All these aspects make discrete geometry, and more specifically arithmetic geometry, a material of choice for new studies and algorithms for image visualization and manipulation. Therefore, we present in the following sections making up the first half of this work a contribution to the field as a collection of results whose common denominator is their intended application to the medical visualization applications presented in the second half of this work.

1 Introduction

First, fundamental notions and selected existing results of discrete geometry are introduced with particular focus on arithmetic digital lines and planes. Then, we tackle the problem of 3D digital lines for which we propose an original approach and derive interesting new results such as the existence of combinatorially distinct 3D digital lines having the same direction. Unlike 2D digital lines that have received considerable attention and for which numerous results have been established, 3D digital lines remain a mostly unexplored area. We present a new method for digitizing Bézier curves and surface patches that is compatible with existing results of digital geometry. Earlier methods generally rely on the choice of an arbitrary constant whose value with respect to the sampling grid is not clear, our approach avoids this problem and approximates Bézier curves and surface patches by digital straight line segments and digital plane pieces in a close to optimal manner. The first half of this work is then concluded with a study of two problems related to what we call *multi-scale discrete geometry*: rectangular sub-lattices of $\mathbb{Z}^2$ induce a tessellation of the plane that can be seen as another scale of discreteness beyond the fundamental tessellation generated by $\mathbb{Z}^2$ itself. The relations between those two levels of discreteness have a theoretical and also practical interest for some applications. In this context we propose a solution to two particular problems: determining the covering of digital lines and of digital parallelograms by regular rectangular tessellations of the plane.

2 2D Digital Lines and Planes

This chapter introduces the basic notions of discrete geometry that make up the foundations of the subsequent chapters. The definitions of digital straight lines and planes as well as their basic properties are presented. Closing the chapter are two original results about the intersection of digital lines and the relation between the arithmetic offset and the induced shift in their combinatorial structure.

2.1. Notations and definitions

In this section we recall some very basic results of number theory and introduce the notations that will be used throughout this work. Proofs for results mentioned here can be found in a reference book on the theory of numbers such as [29].

a,b being two integers, we denote with $\left\lfloor \frac{a}{b} \right\rfloor$ the quotient of the euclidean division of a by b while $\left\{ \frac{a}{b} \right\}$ denotes the remainder of this division, otherwise called residue of a to modulus b. The fundamental relation between these two values writes:

$$a = b\left\lfloor \frac{a}{b} \right\rfloor + \left\{ \frac{a}{b} \right\} \tag{2-1}$$

We denote with $\gcd(a, b)$ the greatest common divider of a and b. a and b are said to be *relatively prime* if $\gcd(a, b) = 1$.

Theorem 2-1. *Let* $(a, b) \in \mathbb{Z}^2$ *and let* $g = \gcd(a, b)$ *then there exist two integers u and v such that:*

$$au + bv = g \tag{2-2}$$

Proof. The demonstration of this fundamental theorem can be found in any treatise on the theory of numbers such as [29]

Often demonstrations will be made with restrictions on the directions of discrete objects for the sake of clarity. Unless otherwise noted, 3D discrete objects have a direction given by an integer vector (a, b, c) verifying the following hypotheses:

$$\begin{cases} \gcd(a, b, c) = 1 \\ 0 \le a < b < c \end{cases} \tag{2-3}$$

The first one, motivated by arithmetical reasons, is not a real restriction: the general case can be quite easily reduced to it. The second one is more interesting as it stems from the symmetries of the $\mathbb{Z}^3$ space (or those of the cube). These inequalities describe what is called the *standard simplex* (Figure 2-1), which is the fundamental domain of the group of symmetries of the cube.

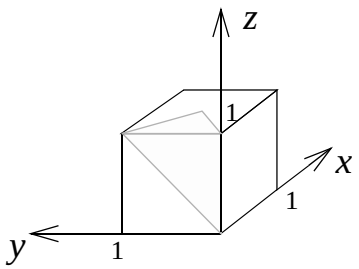


Figure 2-1: The standard simplex

Using this group, the study of 3D digital lines or planes directed by any vector (a, b, c), can be reduced to those directed by vectors belonging to this fundamental domain. Results for the other directions can be easily derived using the symmetries of the cube (combinations of reflections and coordinates swaps).

We also recall here some basic notions of 2D and 3D discrete topologies [35]. Let $\mathbb{Z}$ be the set of relative integers. A point P of $\mathbb{Z}^2$ is defined by its coordinates $(x, y) \in \mathbb{Z}^2$, a point of $\mathbb{Z}^3$ is defined by its coordinates $(x, y, z) \in \mathbb{Z}^3$. We call 2D (resp. 3D) image array, a tessellation of $\mathbb{R}^2$ (resp. $\mathbb{R}^3$) into regular tiles which we call *pixels* (resp. *voxels*). There are two usual conventions to define this tessellation:

- *centered pixels* (resp. voxels): the pixel (resp. voxel) of coordinates (x, y) (resp. (x, y, z)) corresponds to the interval of $\mathbb{R}^2$ (resp. $\mathbb{R}^3$), $[x - 0.5, x + 0.5) \times [y - 0.5, y + 0.5)$ (resp. $[x - 0.5, x + 0.5) \times [y - 0.5, y + 0.5) \times [z - 0.5, z + 0.5)$)
- *edged pixels* (resp. voxels): the pixel (resp. voxel) of coordinates (x, y) (resp. (x, y, z)) corresponds to the interval of $\mathbb{R}^2$ (resp. $\mathbb{R}^3$), $[x, x + 1) \times [y, y + 1)$ (resp. $[x, x + 1) \times [y, y + 1) \times [z, z + 1)$)

These conventions are rigorously equivalent. They induce however some modifications in algebraic equations of discrete objects defined relatively to continuous objects. For instance, the equation of the digital line that covers an euclidean line is not the same if centered or edged pixels are considered (Figure 2-2). For consistence, we always use the "edged pixels" convention.

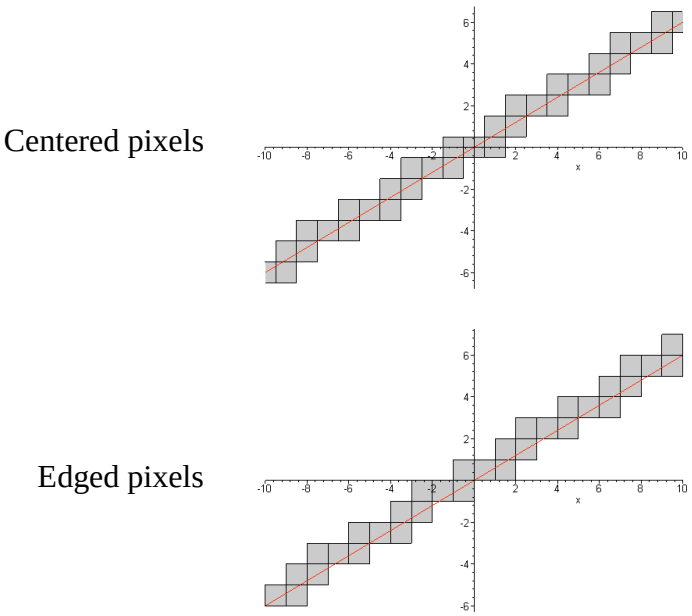


Figure 2-2: Centered vs. edged pixels

The same digital line $D(3,-5,-4,8)$ is drawn using centered and edged pixels. In the first case it covers the euclidean line $3x-5y=0$, in the second it does not.

These tessellations establish a one-to-one correspondence between the integer lattice $\mathbb{Z}^2$ (resp. $\mathbb{Z}^3$) and the elements of a 2D (resp. 3D) image array. We will use these two points of view indifferently and speak either of points or pixels/voxels.

2.1.1. Discrete adjacency and connectedness

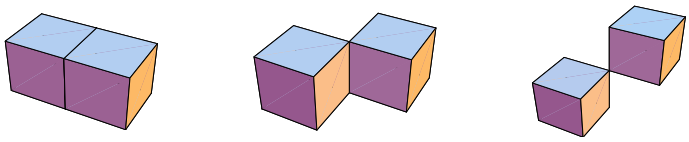
Two distinct points $P_1(x_1,y_1)$ and $P_2(x_2,y_2)$ of $\mathbb{Z}^2$ are said to be

- *8-adjacent* if their coordinates differ by at most 1, $max(|x_2-x_1|, |y_2-y_1|) \leq 1$, i.e. the corresponding pixels share one common vertex.
- *4-adjacent* if one at most of their coordinates differs by 1, $|x_2-x_1| + |y_2-y_1| \leq 1$, i.e. the corresponding pixels share one common edge.

It is clear that two points that are 4-adjacent are also 8-adjacent. We say that two points are *strictly 8-adjacent* when they are 8-adjacent but not 4-adjacent.

Similarly, two distinct points $P_1(x_1, y_1, z_1)$ and $P_2(x_2, y_2, z_2)$ of $\mathbb{Z}^3$ are said to be

- *26-adjacent* if their coordinates differ by at most 1, $max(|x_2 - x_1|, |y_2 - y_1|, |z_2 - z_1|) \leq 1$, i.e. the corresponding voxels share one common vertex.
- *18-adjacent* if two at most of their coordinates differ by 1, i.e. the corresponding voxels share one common edge.
- *6-adjacent* if one at most of their coordinates differs by 1, $|x_2 - x_1| + |y_2 - y_1| + |z_2 - z_1| \leq 1$, i.e. the corresponding voxels share one common face.



6-adjacency 18-adjacency 26-adjacency

Figure 2-3: Discrete adjacencies in 3D

It is also clear in this case that if two points are 6-adjacent then they are also 18-adjacent and 26-adjacent. We say two points are *strictly 26-adjacent* if they are 26-adjacent but not 18-adjacent. We say they are *strictly 18-adjacent* if they are 18-adjacent but not 6-adjacent.

For $n = (4, 6)$ respectively $(6, 18, 26)$, the *n-neighbors* of a point P are the n points that are n-adjacent to P. The set of n neighbors to a point defines its *n-neighborhood*.

A *n-path* is an ordered set of points such that consecutive pairs are n-adjacent. A set S of points is said to be *n-connected* if there exists an n-path in S between every pair of points of S. In $\mathbb{Z}^2$, a 4-connected set is also 8-connected. We say a set is *strictly 8-connected* if there exists a pair of points in that set that can be linked by an 8-path but not by a 4-path contained in the set. Similarly in $\mathbb{Z}^3$, a 6-connected set is also 18-connected and 26-connected. We say that a set is *strictly 18-connected* (resp. *strictly 26-connected*) when there exists a pair of points in the set that can be linked by an 18-path (resp. 26-path) but not by a 6-path (resp. an 18-path) entirely contained in the set.

2.1.2. Discrete tunnels

Let S be a subset of a set of points X, if $X - S$ is not n-connected then S is said to be *n-separating* in X [11]. If S is n-separating but not m-separating for $m > n$ then S is said to have *m-tunnels* (Figure 2-4). If S is 26-separating then it is said to be tunnel-free. Discrete tunnels in curves and surfaces are often a problem since they tend to defeat the Jordan theorem.

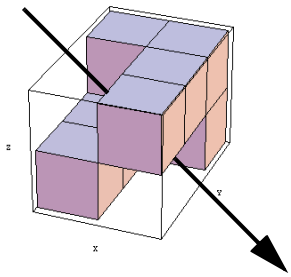


Figure 2-4: A 18-tunnel through a piece of a 6-separating surface

2.1.3. The Jordan theorem for discrete curves

In $\mathbb{R}^2$, the *Jordan theorem* is a fundamental theorem which states that any simple closed curve divides $\mathbb{R}^2$ into two disjoint connected components, one finite is called the *interior* of the curve, the other, infinite, is called the *exterior* of the curve (Figure 2-5).

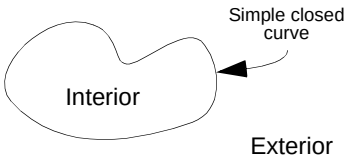


Figure 2-5: The Jordan theorem in $\mathbb{R}^2$

A discrete *n-simple closed curve* is defined as a connected set S of points each of which is n-adjacent to exactly two other points in the set. In $\mathbb{Z}^2$, the Jordan theorem still holds provided two different types of connectivity are considered for the curve and the background, otherwise contradictions appear [35]. One can see for instance in the first figure of Figure 2-6 that if 8-

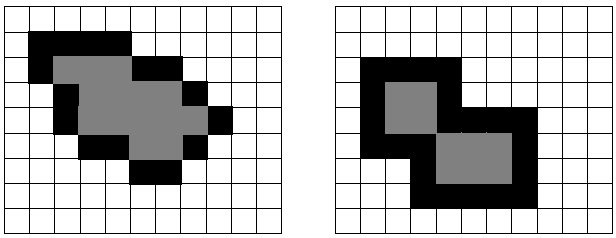


Figure 2-6: The Jordan theorem in $\mathbb{Z}^2$

connectivity was to be considered for both the curve and the background then the white and gray voxel sets which are the expected exterior and interior of the curve would be connected, which is in contradiction with the theorem. Now if 8-connectivity is considered for the black points

defining the curve and 4-connectivity for the other points, then the sets defined by gray and white pixels are disjoint and thus one can speak of interior and exterior of the curve in the sense of the Jordan theorem.

Unfortunately this result is very hard to extend to $\mathbb{Z}^3$ and today researchers still spend a lot of energy in finding a satisfactory general framework for discrete surfaces in which the Jordan theorem holds. Some demonstrations have been made though by Herman [30], Malgouyres [42] and others (references in [20]). Unfortunately those demonstrations rely more or less on ad hoc definitions of surfaces which clearly shows that the real problem resides in the very definition of a discrete surface. Indeed, it has been shown that unlike continuous simple closed surfaces, thin discrete simple closed surfaces verifying the Jordan theorem cannot be characterized locally [5], therefore a variety of approaches for defining surfaces have been made, some of which are compatible with the Jordan theorem: a voxel approach, where a surface is defined as a set of faces (surfels) between adjacent pairs of voxels [30], a combinatorial approach, where a surface is defined as a combinatorial manifold [20], a graph-theoretical approach where a surface is defined as a thin set of points linked by adjacency relations and additional properties [5], [42].

2.2. Digital straight lines (previous art)

2.2.1. Introduction and definition

Straight lines are the most fundamental objects in computer graphics just as they are in euclidean geometry. It is no surprise then that many researchers focused some interest on their study. Digital lines are indeed a fascinating subject as, despite their apparent simplicity, they carry much of the still somewhat mysterious relations between the discrete and continuous.

The common way of thinking geometry is so deeply pervaded by continuous concepts that the most intuitive approach to digital straight lines is naturally, to consider them as approximations of euclidean lines. This first approach is fundamental and was widely investigated [2], [36], [48]. It lead for instance, to the first drawing algorithms of lines on a computer display [6]. Thus the very first definitions of digital straight line segments were actually algorithmic rather than properly geometric. Rosenfeld first formulated the fundamental chord property which characterizes digital straight line segments [48] and deduced their first important intrinsic properties. Most of this early work however remained strongly rooted in euclidean geometry.

Relying on continuous geometry to study digital lines is certainly reassuring and leads to many interesting results but it also uncovers many problems (e.g., the intersection of lines) and actually somewhat neglects the fundamental nature of digital lines. An alternative approach is to consider digital lines for what they are, i.e., discrete sets of discrete points, define them as such and deduce properties from this definition. Among the works in that direction, the arithmetic definition proposed by Reveillès [46] is certainly the most fruitful and promising. It

achieves a major step in the synthesis of existing definitions into a remarkably simple and rich formulation.

Definition 2-1. We call *planar digital straight line* a subset of $\mathbb{Z}^2$ described by a diophantine equation of the following form :

$$D(a, b, \gamma, \rho) = \{(x, y) \in \mathbb{Z}^2 / \gamma \leq ax + by < \gamma + \rho\} \qquad (2\text{-}4)$$

where all parameters are integers. (a, b) defines the direction of the line, γ defines its *affine offset* while ρ is called *arithmetical thickness*.

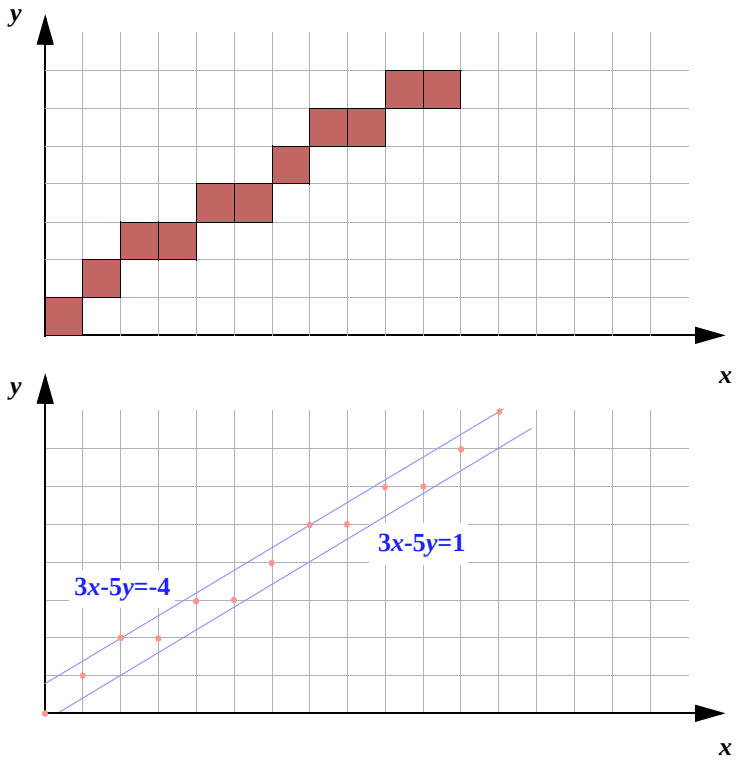


Figure 2-7: The digital line $D(3, -5, -4, 5)$

A digital line can be seen as either a set of pixels or equivalently as a set of integer points comprised between two real euclidean lines. The same line is shown here using both representations.

When considered in $\mathbb{R}^2$ instead of $\mathbb{Z}^2$, Equation 2-4 defines a continuous strip of width (measured orthogonally to the direction of the strip) $w = \frac{\rho}{\sqrt{a^2 + b^2}}$ which can be thought of as the continuous counterpart of the naive digital line. This offers another point of view on digital lines where those are seen as sets of integer points contained in a continuous strip as illustrated in Figure 2-7.

2.2.2. Naive digital lines

A particularly interesting subset of digital lines consists of those verifying $\rho = max(|a|, |b|)$:

$$D(a, b, \gamma) = \{(x, y) \in \mathbb{Z}^2 / \gamma \leq ax + by < \gamma + max(|a|, |b|)\} \qquad (2\text{-}5)$$

which are called *naive digital lines* and have exactly the same structure as the sets drawn by the classical Bresenham algorithm for lines [6]. The most important property of naive lines is strict 8-connectivity.

Naive digital lines are also "functional" along one of the main axes. For instance let us suppose that $0 < a < b$, then for any value of x there is one single value of y such that (x, y) belongs to the naive digital line $D(a, b, \gamma)$, i.e., y can be written as a function of x, hence the term "functional". More precisely, in that case:

$$y = -\left\lceil \frac{ax - \gamma}{b} \right\rceil \qquad (2\text{-}6)$$

This fact can be further understood by noticing that in the case of naive digital lines such that $0 < a < b$, the height of the continuous strip defined in $\mathbb{R}^2$ by Equation 2-5 is exactly 1, which guarantees that above any integer abscissa there is one and one only integer point contained in that continuous strip.

Naturally the previous result still holds by symmetry if $a > b$. In that case the naive digital line is functional along y, meaning that x can be written as a function of y.

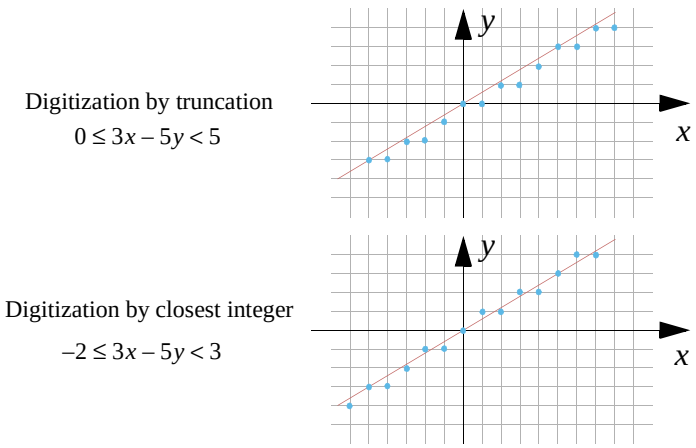


Figure 2-8: Digitization by truncation vs. digitization by closest integer

$D(a, b, \gamma)$ is the digitization by truncation of the ordinary euclidean line of equation $ax + by = \gamma$ where x, y, z are real numbers and a, b and γ are integers, while its digitization by the closest integer point is given by $D\left(a, b, \gamma - \left\lceil \frac{max(|a|, |b|)}{2} \right\rceil\right)$ (Figure 2-8).

2.3. Digital planes (previous art)

2.3.1. Introduction and definition

One of the particularly interesting aspects of Definition 2-1 is that it extends particularly well to describe digital planes [16]. Therefore digital planes share properties similar to those of digital lines.

Definition 2-2. We call *digital plane* a subset of $\mathbb{Z}^3$ described by a diophantine equation of the following form :

$$P(a, b, c, \gamma, \rho) = \{(x, y, z) \in \mathbb{Z}^3 / \gamma \leq ax + by + cz < \gamma + \rho\} \qquad (2\text{-}7)$$

$(a, b, c) \in \mathbb{Z}^3$ defines the normal direction of the plane, $\gamma \in \mathbb{Z}$ defines its *affine offset* while $\rho \in \mathbb{Z}$ is called *arithmetical thickness*.

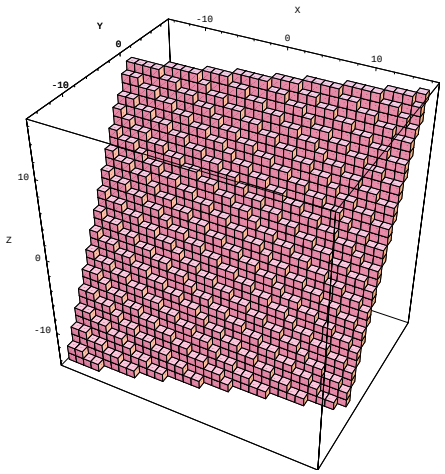


Figure 2-9: A naive digital plane

2.3.2. Naive digital planes

As for digital lines, a particularly interesting subset of digital planes consists of those verifying $\rho = max(|a|, |b|, |c|)$:

$$P(a, b, c, \gamma) = \{(x, y, z) \in \mathbb{Z}^3 / \gamma \leq ax + by + cz < \gamma + max(|a|, |b|, |c|)\} \qquad (2\text{-}8)$$

which are called *naive digital planes*. Those planes are 18-connected and have no holes for 6-connectivity. One of the most fundamental properties of digital planes is that they are "functional" along the main direction of their normal, i.e., if $(c = max(|a|, |b|, |c|)) > 0$ then for each (x, y) there exists a single z such that (x, y, z) belongs to the digital plane. That is, z can be expressed as a function of (x, y), which writes:

$$z = -\left\lfloor \frac{ax + by - \gamma}{c} \right\rfloor \qquad (2\text{-}9)$$

$P(a, b, c, \mu)$ is the digitization by truncation of the ordinary euclidean plane of equation $ax + by + cz = \mu$ where x, y, z are real numbers and a, b and μ are integers. $P(a, b, c, \mu)$ also represents the digitization by the closest integer point of the plane $\alpha x + \beta y + \gamma z = \delta$ where $\alpha = a/c$, $\beta = b/c$, $\gamma = 1$, $\delta = d/c$ and $\mu = d - \left\lfloor \frac{c}{2} \right\rfloor$.

2.4. Basic properties of digital lines and planes (previous art)

Various properties of digital lines can be deduced from Definition 2-1 [46], to summarize a few:

- The structure of a digital line of direction (a,b) with $0 < a < b$ and $\gcd(a, b) = 1$ is described by the modular sequence $\left\{ \left\lfloor \frac{ai}{b} \right\rfloor \right\}_{0 \leq i < b}$. The structure of a digital line is therefore b-periodic and this modular sequence describes the length of the plateaux of the line and their sequence.
- Two digital lines having the same direction and the same arithmetical thickness are equivalent, i.e., are identical within translation
- Two digital lines having the same bounds are homologous, i.e., one can be transformed into the other by a unimodular matrix (which represents a sequence of shearing operations)
- The thickness parameter ρ controls the connectivity of the line :
  - $\rho < max(|a|, |b|)$: the line is disconnected
  - $\rho = max(|a|, |b|)$: the line is strictly 8-connected

- $\rho = |a| + |b|$: the line is strictly 4-connected
- $\rho > |a| + |b|$: the line is thick

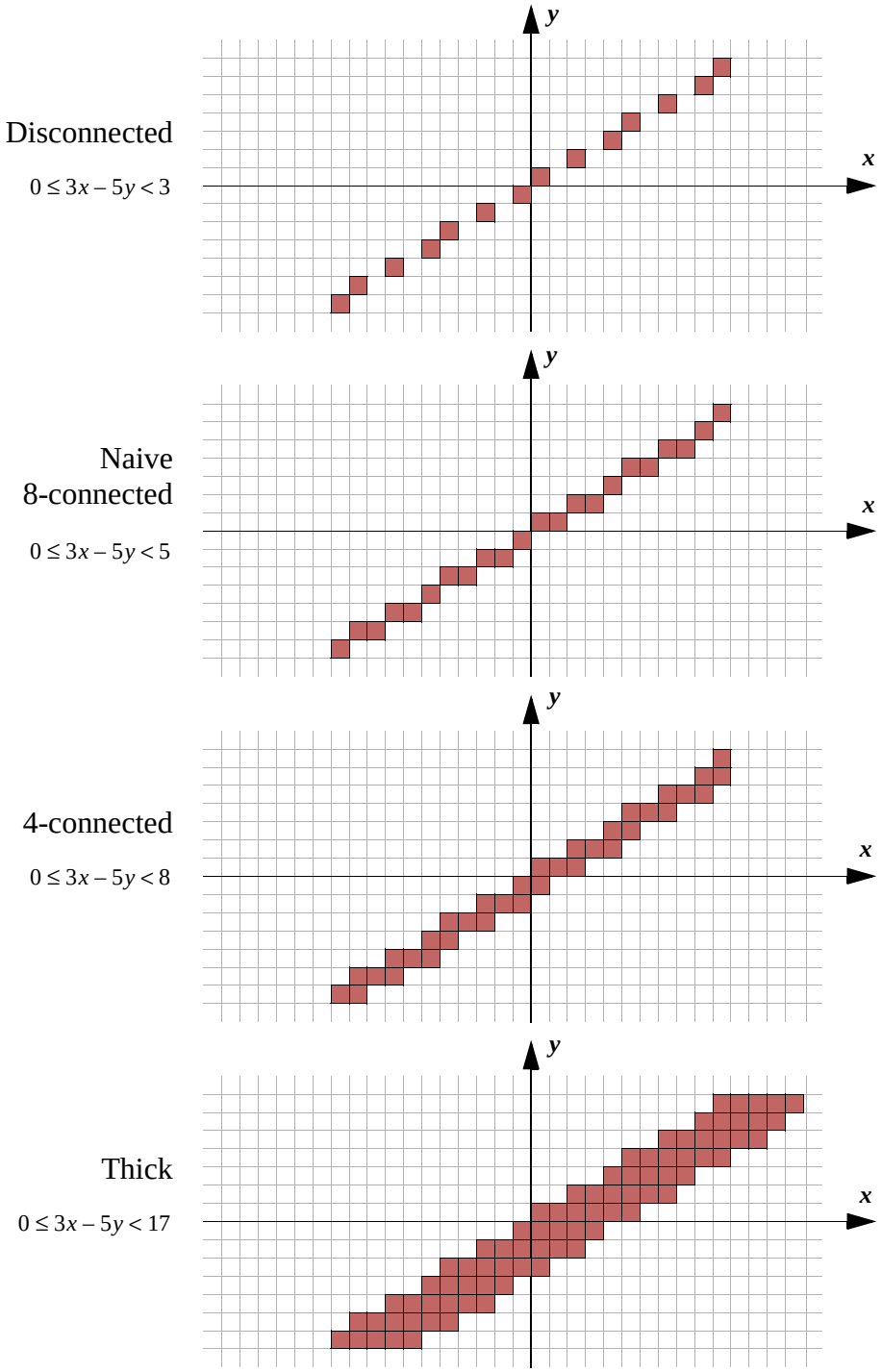


Figure 2-10: Connectivity vs. arithmetical thickness of digital lines

These properties also apply to digital planes. In particular, in the case of a digital plane:

- The thickness parameter ρ controls the connectivity of the plane :
  - $\rho < max(|a|, |b|, |c|)$: the plane has 6-holes
  - $\rho = max(|a|, |b|, |c|)$: the plane is strictly 18-connected and has no 6-holes
  - $\rho = |a| + |b| + |c|$: the plane is 6-connected
  - $\rho > |a| + |b| + |c|$: the plane is thick

2.5. New results about digital lines

The following sections present original results about problems related to 2D digital lines. The first one deals with the intersection of 2D digital lines, generalizing a result previously established by Reveillès [46]. Such a generalization is necessary for practical implementations which generally need to avoid restrictions on the directions of the lines. The algorithm that computes the intersection of digital lines is fundamental, for instance, it constitutes the basis of a digital parallelogram drawing algorithm.

The second result introduced in this section links the affine offset parameter γ of the equation of a digital line to a shift of index in the canonical combinatorial pattern of the line. Indeed it has been shown that the combinatorial structure of a naive line, i.e., its sequence of steps and plateaux, depends only on its direction [46]. However varying the affine offset induces a shift in the canonical pattern. The relation between these two values can be used for instance to devise an optimized naive digital line drawing algorithm.

2.5.1. Intersection of 2D digital lines with non prime direction coefficients

Most of the time a requirement is made on the direction of digital lines that the coefficients be mutually prime. Indeed this is not much of a restriction from a theoretical point of view and greatly simplifies the calculations but it also happens to be somewhat inconvenient in practice when one tries to implement algorithms. The intersection of 2D digital lines has been studied by Reveillès [46] with such restrictions, here we extend the results to digital lines of arbitrary rational direction. The intersection of discrete lines can be very complex and in particular it may not be connected.

Let $D_1(a, b, \gamma, \rho)$ and $D_2(c, d, \mu, \nu)$ be two digital lines. We assume that the lines are not parallel: $ad - bc \neq 0$. In order to find their intersection we need to solve the following equation system:

$$\begin{cases} \gamma \leq ax + by < \gamma + \rho \\ \mu \leq cx + dy < \mu + \nu \end{cases} \quad (2\text{-}10)$$

Let $g = gcd(a, b)$, $a = ga'$ and $b = gb'$. Then Equation 2-10 can be rewritten as (using matrix notation):

$$\begin{bmatrix} \frac{\gamma}{g} \\ \mu \end{bmatrix} \leq \begin{pmatrix} a' & b' \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} < \begin{bmatrix} \frac{\gamma + \rho}{g} \\ \mu + \nu \end{bmatrix} \tag{2-11}$$

Since a' and b' are mutually prime, there exist u and v such that $a'u + b'v = 1$.

We introduce $U = \begin{pmatrix} u & -b' \\ v & a' \end{pmatrix}$ and the change of coordinates $\begin{pmatrix} X \\ Y \end{pmatrix} = U^{-1} \begin{pmatrix} x \\ y \end{pmatrix}$. Equation 2-11 becomes

$$\begin{bmatrix} \frac{\gamma}{g} \\ \mu \end{bmatrix} \leq \begin{pmatrix} 1 & 0 \\ cu + dv & a'd - b'c \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} < \begin{bmatrix} \frac{\gamma + \rho}{g} \\ \mu + \nu \end{bmatrix} \tag{2-12}$$

Let $\lambda = cu + dv$ and $\delta = a'd - b'c$. The solution for the intersection can be computed with a double loop in X and Y such that

$$-\left\lceil \frac{-\gamma}{g} \right\rceil \leq X < -\left\lceil \frac{-\gamma - \rho}{g} \right\rceil \tag{2-13}$$

The expression for the boundaries of Y depends on the sign of δ:

- $\delta > 0$

$$-\left\lceil \frac{-\mu + \lambda X}{\delta} \right\rceil \leq Y < -\left\lceil \frac{-\mu - \nu + \lambda X}{\delta} \right\rceil \tag{2-14}$$

- $\delta < 0$

$$\left\lceil \frac{\mu + \nu - \lambda X}{\delta} \right\rceil + 1 \leq Y < \left\lceil \frac{\mu - \lambda X}{\delta} \right\rceil + 1 \tag{2-15}$$

Example. Let us apply these results to find the intersection of $D_1(4, -10, 5, 15)$ and $D_2(6, -18, 4, 18)$:

$$g = gcd(4, 10) = 2 \tag{2-16}$$

Hence $3 \le X < 10$, furthermore $\delta = -6$ and therefore, the value of Y is given by Equation 2-15:

$$\left\lfloor \frac{6X - 22}{6} \right\rfloor + 1 \le Y < \left\lfloor \frac{6X - 4}{6} \right\rfloor + 1 \qquad (2\text{-}17)$$

Finally, applying the unimodular matrix $U = \begin{pmatrix} -2 & 5 \\ -1 & 2 \end{pmatrix}$ on (X, Y) yields the final result shown in Figure 2-11 and Figure 2-12.

X	3			4			5			6			7			8			9		
Y	0	1	2	1	2	3	2	3	4	3	4	5	4	5	6	5	6	7	6	7	8
x	-6	-1	4	-3	2	7	0	5	10	3	8	13	6	11	16	9	14	19	12	17	22
y	-3	-1	1	-2	0	2	-1	1	3	0	2	4	1	3	5	2	4	6	3	5	7

Figure 2-11: Intersection of $D_1(4, -10, 5, 15)$ and $D_2(6, -18, 4, 18)$

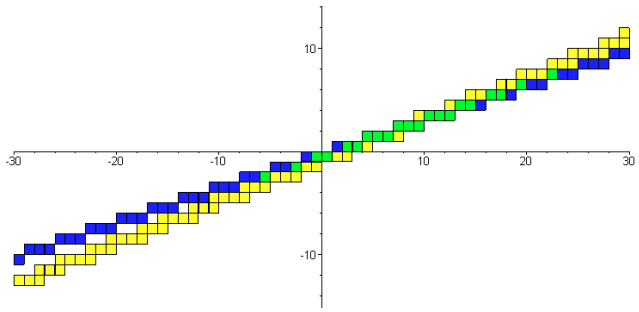


Figure 2-12: Intersection of $D_1(4, -10, 5, 15)$ and $D_2(6, -18, 4, 18)$

2.5.2. Index shift in the canonical pattern of a 2D discrete line

As previously mentioned (Section 2.4) a 2D digital line of equation $\gamma \le ax - by < \gamma + \rho$ such that $0 < a < b$ and $\gcd(a, b) = 1$ is b-periodic, i.e. it can be built by repeating periodically a pattern of b pixels (Figure 2-13). An optimized line drawing algorithm can take advantage of that result by computing the canonical pattern once and then repeating it.

It is easy to show that the digital line pattern depends only on the direction coefficients $(a, -b)$. Varying the affine offset γ has no influence on the line pattern since digital lines that have the same direction but different affine offsets are identical within integer translation (Figures 2-13 and 2-14). Thus the line pattern can be calculated independently of the affine offset.

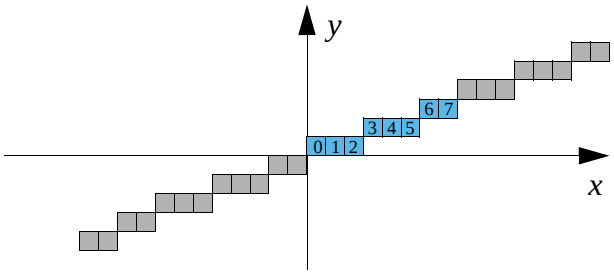


Figure 2-13: The canonical pattern of $D(3, -8, 0)$

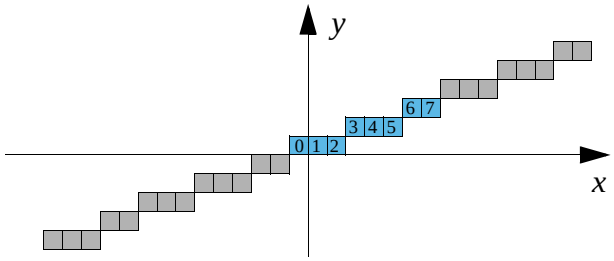


Figure 2-14: $D(3, -8, -3)$ **is equal to** $D(3, -8, 0)$ **within translation**

We define the canonical pattern of a direction $(a, -b)$ as the sequence (x_i, y_i) for $0 \le i < b$, where

$$\begin{cases} x_i = i \\ y_i = \left\lfloor \frac{ai}{b} \right\rfloor \end{cases} \tag{2-18}$$

To draw a digital line $\gamma \le ax - by < \gamma + \rho$ starting at $x = 0$, a line drawing algorithm using the previous optimization needs to determine the shift in the canonical pattern induced by $\gamma \ne 0$. Figure 2-14 shows, for instance, that at $x = 0$ the index shift of $D(3, -8, -3)$ in the canonical pattern for direction $(3, -8)$ is 1.

Theorem 2-2. *$D(a, -b, \gamma)$ can be drawn by shifting the canonical pattern for direction $(a, -b)$ s units forward at $x = 0$ where*

$$s = \left\{ \frac{-\gamma u}{b} \right\} \tag{2-19}$$

Proof. The structure of the canonical pattern is controlled by the modular sequence $\left\{\frac{ai}{b}\right\}$ while the structure of $D(a, -b, \gamma)$ is described by $\left\{\frac{ai-\gamma}{b}\right\}$, thus we are looking for the shift $0 \le s < b$ such that:

$$\left\{\frac{ai-\gamma}{b}\right\} = \left\{\frac{a(i+s)}{b}\right\} \tag{2-20}$$

Since $\gcd(a, b) = 1$ there exist (u, v) such that $au + bv = 1$. Hence we can write:

$$\left\{\frac{ai-\gamma}{b}\right\} = \left\{\frac{ai-\gamma(au+bv)}{b}\right\} = \left\{\frac{ai+a(-\gamma u)}{b}\right\} = \left\{\frac{a\left(i+\left\{\frac{-\gamma u}{b}\right\}\right)}{b}\right\} \tag{2-21}$$

■

3 Theoretical Aspects of 3D Digital Lines

In this chapter, a new approach to the study and definition of 3D digital lines is proposed. This approach based on the properties of the rational lattice generated from the projection of $\mathbb{Z}^3$ onto an euclidean plane of rational direction opens the door to interesting new results and properties of digital lines.

3.1. Introduction

Research work about 3D digital lines has followed two different paths, much like 2D digital lines. The first one focused on algorithmics while the second one investigated more theoretical results such as the geometrical and topological properties of 3D discrete lines.

Especially because of the increasing popularity of ray tracing for the rendering of 3D scenes and the development of optimization techniques like space partitioning into voxels, the need for 3D line drawing algorithms has become urgent. The first attempts derived directly from euclidean geometry with little or no attention paid to the discrete nature and properties of the objects being built [2]. Then the classical 2D line drawing algorithms such as Bresenham's were extended to the 3D case by considering two projections of the line onto the main coordinates planes [32]. This approach which brought a significant improvement over the previous algorithms by using integer arithmetic, has proved to be particularly convenient and remains one of the most commonly used even though more recent algorithms such as Cohen and Kaufman's Tripod Algorithm have introduced a new efficient way of drawing 6-connected 3D digital lines [13].

Regarding the theoretical aspects, early research tried to extend the knowledge and theoretical results on 2D digital lines to investigate the structure and properties of 3D digital lines. C.E. Kim for instance showed that one could characterize a 3D digital line thanks to the chord property applied to its projections onto the coordinate planes [33]. She also demonstrated that the chord property does not hold for the digital line itself, thus showing that the mathematical study of discrete geometric objects becomes much more intricate in 3D. However this definition of 3D digital lines using the 2D digital lines of closest integer points in two of the projections, has several limitations:

- the discrete topology of this 3D digital line notion is not clear,
- its third projection is, generally, not the closest set of points of the third euclidean projection,
- if we consider a family of parallel euclidean lines, we do not know how many combinatorially distinct digital structures will be built by this process,

- and above all the set of voxels defined in this way is not the set of closest points of the given 3D euclidean line (see Figure 3-1).

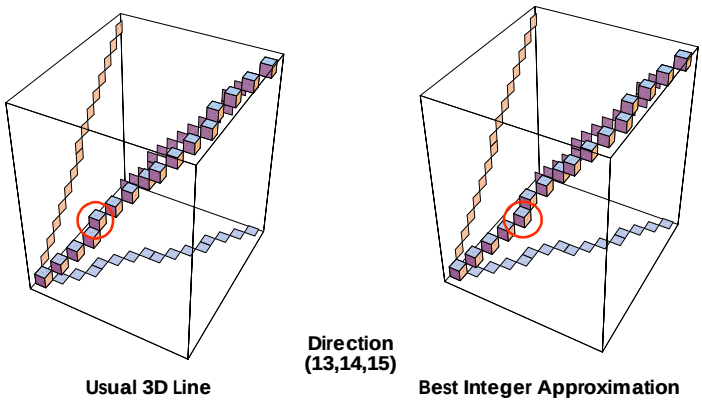


Figure 3-1: Usual 3D line vs. best integer approximation

The graphics on the left shows a digital 3D line built in the usual way using the Bresenham's algorithm for two of its projections. The graphics on the right shows the discrete line built with the integer points that are closest to the corresponding euclidian line. The red circles shows where a difference appears.

These questions are only the simplest ones. Many others could be asked such as: the dependence on the choice of the projections, the intersections with digital planes, the intersections between 3D digital lines, ... Thus a new approach is needed. Therefore we would like to provide here a new definition of 3D digital lines relying on subgroups of $\mathbb{Z}^3$, whose main advantage over the former one is its ability to convert any practical question into rigorous algebraic terms. In particular, we obtain a complete description of the topology of these lines and a condition for the third projection being a 2D digital line as well as a classification of digital lines of the same direction into classes of equivalent combinatorial structure. The approach presented here offers new perspectives and looks particularly promising, however its in-depth exploitation would have fallen much beyond the scope of this particular work and the results we present here are to be considered more as hints towards new directions for the study of digital lines rather than complete results per se.

In the second part of this section we use the arithmetical definition of 3D digital lines derived from this approach to study the intersection between a digital line or a set of adjacent digital lines and a digital plane.

3.2. The projection lattice of the integers along a rational direction

The general idea behind our approach consists in studying the properties of the rational lattice generated on an euclidean plane by the orthogonal projection of $\mathbb{Z}^3$ onto that plane. For the sake

of clarity in the demonstrations, we present the definition with a restriction to directions given by integer vectors (a, b, c) satisfying the hypotheses of Equation 2-3.

Let us consider the *euclidean plane* P, normal to (a, b, c) whose equation is

$$ax + by + cz = 0 \tag{3-1}$$

and the orthogonal projection π of $\mathbb{Z}^3$ onto P:

$$\pi: \quad \mathbb{Z}^3 \to P \tag{3-2}$$

It is easy to prove that the image

$$\mathcal{L} = \pi(\mathbb{Z}^3) \tag{3-3}$$

is a *discrete* and rational lattice.

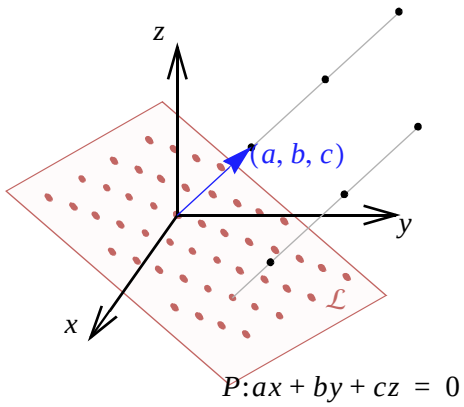


Figure 3-2: Projection of $\mathbb{Z}^3$ onto a plane along a rational direction

Brown dots are the projections of integer points of $\mathbb{Z}^3$ and make up the lattice $\mathcal{L}$. Two fibers of $\mathbb{Z}^3$ are shown.

An important consequence results from $\mathcal{L}$ discreteness: bounded subsets of plane P contain only a *finite* numbers of points of $\mathcal{L}$. If B is such a bounded set, its inverse image $\pi^{-1}(B)$ is made of a finite number of *fibers* all of which are in one to one correspondence with the subgroup

$$\pi^{-1}(0) = \{k \cdot (a, b, c) \mid k \in \mathbb{Z}\} \tag{3-4}$$

generated by vector (a, b, c) (see Figure 3-2). More precisely for any point $x \in \mathcal{L}$ we know that its fiber $\pi^{-1}(x)$ is equal, within translation, to $\pi^{-1}(0)$.

In this way we reduce the study of 3D digital lines to the study of the 2D lattice $\mathcal{L}$. Lattices (or $\mathbb{Z}$-modules) are structures which are, at the same time, similar and distinct from vector spaces. The reader will find their properties in any algebra treatise [38].

We are more precisely interested in the study of the subset $\mathfrak{p}$ of $\mathcal{L}$ contained in a fundamental domain of a sub-lattice $\mathcal{S}$ of $\mathcal{L}$. We show hereafter that the parameterization of $\mathcal{L}$ and $\mathfrak{p}$ can be made extremely simple thus leading to a particularly interesting representation of digital 3D lines. We show also that we can, among others, read the topology of the line and recover usual algorithms from this representation.

3.2.1. Simplification of triply generated two dimensional lattices

The main difficulty concerning $\mathbb{Z}$-modules (or lattices or free abelian groups) is that one can find *free families* of vectors whose cardinal is equal to the dimension of the ambient space and which *do not generate* this space. One such example is given by the set $\{(1, 0), (3, 2)\}$ of $\mathbb{Z}^2$, which is free, has cardinal two and is not generator of $\mathbb{Z}^2$. We can see for instance that vector $(2, 1)$ cannot be represented as a linear combination, with *integer* coefficients, of the given vectors.

Let us introduce the following definitions:

- If $\mathbf{V_1}, \mathbf{V_2}, \ldots, \mathbf{V_i}$ are integer vectors of $\mathbb{Z}^3$, $L(\mathbf{V_1}, \mathbf{V_2}, \ldots, \mathbf{V_i})$ will denote the lattice generated by these vectors. We shall restrict to $i = 2$ and $i = 3$ and say, respectively, in these cases that the lattices are *doubly* or *triply* generated.
- We shall denote any fundamental domain of $L(\mathbf{V_1}, \mathbf{V_2})$ by Par (for parallelogram) and by $\mathfrak{p}$ the set of points of $L(\mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3})$ contained in Par, (keep in mind that the points $\mathbf{V_1}, \mathbf{V_2}$ and $\mathbf{V_1} + \mathbf{V_2}$ are *not* members of $\mathfrak{p}$.
- We also denote by $\nu(\mathfrak{p})$ the cardinal of $\mathfrak{p}$.

Our goal is parameterize in the most simple way the set $\mathfrak{p}$. Let us start with a 2D version of this problem, that is we suppose vectors $\mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3}$ are in $\mathbb{Z}^2$.

$\mathbf{V_1} = (x_1, y_1)$ and $\mathbf{V_2} = (x_2, y_2)$ either generate a rank one subgroup, if $\det\begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = 0$, or a rank two subgroup of $\mathbb{Z}^2$ otherwise. We suppose this last hypothesis is satisfied in what follows.

If $\mathbf{V_3} = (x_3, y_3)$ is a third vector of $\mathbb{Z}^2$ we also suppose that:

- $\gcd(x_i, y_i) = 1$ for $i = 1, 2, 3$ and that
- All three determinants $\det(V_1, V_2)$, $\det(V_1, V_3)$ and $\det(V_3, V_2)$ are non zero and that the first one is positive

With these hypotheses the lattice $L(\mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3})$ is a rank two submodule of $\mathbb{Z}^2$ and we are interested, as explained above, in the set $\mathfrak{p}$ of its points contained in a fundamental domain, Par, of $L(\mathbf{V_1}, \mathbf{V_2})$.

Obviously the parallelogram Par and its translations by the vectors $k_1 \cdot \mathbf{V_1} + k_2 \cdot \mathbf{V_2}$, $k_1, k_2 \in \mathbb{Z}$, induce a tiling of $\mathbb{Z}^2$ (see Figure 3-3). So any integer point of $\mathbb{Z}^2$ belonging to one such tile has a reduction (homologous point) in Par. Thus, after reduction, the sequence $k \cdot \mathbf{V_3}$, $k \in \mathbb{Z}$ gives a subset of the integer points of Par; this inclusion is generally *strict*. In the remaining the following two notations will be used for this reduction modulo Par: either "mod Par" or "mod $\{\mathbf{V_1}, \mathbf{V_2}\}$".

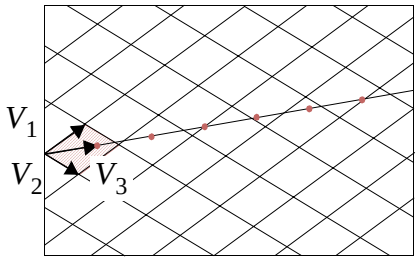


Figure 3-3: A doubly generated lattice and the multiples of a third vector

The integer vectors $\mathbf{V_1}$ and $\mathbf{V_2}$ induce a tiling of $\mathbb{Z}^2$. Multiples of $\mathbf{V_3}$, shown as brown dots, can be reduced to the fundamental parallelogram (hatched area) and therefore describe a subset of the points in that parallelogram.

The following lemma results immediately.

Lemma 3-1. *The set* $\mathfrak{p}$ *is given by the reduction modulo* Par *of the integer multiples of vector* $\mathbf{V_3}$*, that is of* $\{k \cdot \mathbf{V_3} \mid k \in \mathbb{Z}\}$*:*

$$\mathfrak{p} = \{k \cdot \mathbf{V_3} \mid k \in \mathbb{Z}\} \bmod \{\mathbf{V_1}, \mathbf{V_2}\} \tag{3-5}$$

It is well known that the surface of Par is given by $x_1y_2 - x_2y_1$, this quantity $\delta = x_1y_2 - x_2y_1$ also corresponds to the number of integer points contained in Par, i.e. $\mathrm{card}(Par)$. So the cardinal of $\mathfrak{p}$ is bounded by δ. Moreover there exists one integer value k such that $k\mathbf{V_3}$ belongs to the lattice generated by $\mathbf{V_1}$ and $\mathbf{V_2}$. This comes from the following observation: as $\mathrm{card}(Par)$ is *finite*, there must be two distinct integer values m and n such that $m \cdot \mathbf{V_3}$ and $n \cdot \mathbf{V_3}$ have the same reduction mod $\{\mathbf{V_1}, \mathbf{V_2}\}$. Thus for $k = m - n$, $k \cdot \mathbf{V_3}$ is congruent to the null vector:

$$k \cdot \mathbf{V_3} \equiv 0 \bmod \{\mathbf{V_1}, \mathbf{V_2}\} \tag{3-6}$$

which is equivalent to this assertion

It can then be deduced that there exists a smallest non-null integer value, still denoted k, such that $k \cdot \mathbf{V_3}$ belongs to the lattice generated by $\mathbf{V_1}$ and $\mathbf{V_2}$, which can be written as $k\mathbf{V_3} \equiv 0 \bmod \{\mathbf{V_1}, \mathbf{V_2}\}$.

Lemma 3-2. *The smallest non-null integer number k such that $k\mathbf{V_3} \equiv 0 \bmod \{\mathbf{V_1}, \mathbf{V_2}\}$ is exactly the cardinal of $\mathfrak{p}$.*

Proof. Let us consider the elements $\mathfrak{p}_0 \ldots \mathfrak{p}_{k-1}$ of $\mathfrak{p}$, where $\mathfrak{p}_i \equiv i\mathbf{V_3} \bmod \{\mathbf{V_1}, \mathbf{V_2}\}$. These elements are distinct for if we suppose $\mathfrak{p}_m = \mathfrak{p}_n \quad 0 \le m < n \le k$ then we would have $\mathfrak{p}_{n-m} \equiv 0 \bmod \{\mathbf{V_1}, \mathbf{V_2}\}$ which is in contradiction with our hypothesis stating that k is the smallest non-null integer such that $k\mathbf{V_3} \equiv 0 \bmod \{\mathbf{V_1}, \mathbf{V_2}\}$. Therefore $\nu(\mathfrak{p}) \ge k$. Moreover for any integer $q > k$ we have

$$q\mathbf{V_3} \bmod \{\mathbf{V_1}, \mathbf{V_2}\} \equiv \left(k\left\lfloor \frac{q}{k} \right\rfloor + \left\{ \frac{q}{k} \right\} \right)\mathbf{V_3} \bmod \{\mathbf{V_1}, \mathbf{V_2}\} \equiv \mathfrak{p}_{\left\{ \frac{q}{k} \right\}} \tag{3-7}$$

Hence any $q\mathbf{V_3}$ with $q > k$ reduces to one of $\mathfrak{p}_0 \ldots \mathfrak{p}_{k-1}$ within modulo $\{\mathbf{V_1}, \mathbf{V_2}\}$ which allows us to conclude: $k = \nu(\mathfrak{p})$.■

We introduced above the determinant: $\delta = x_1y_2 - x_2y_1$ (which can be supposed to be strictly positive). Let us also consider the values of the two other determinants reduced modulo δ: $\chi \equiv (x_1y_3 - x_3y_1) \bmod \delta$ and $\gamma \equiv (x_3y_2 - x_2y_3) \bmod \delta$

With a little knowledge from group theory about elements order, we can deduce that

Lemma 3-3. *The cardinal of $\mathfrak{p}$, $\nu(\mathfrak{p})$ can be expressed as*

$$k = \nu(\mathfrak{p}) = \mathrm{lcm}\left(\frac{\delta}{\gcd(\chi, \delta)}, \frac{\delta}{\gcd(\gamma, \delta)} \right) \tag{3-8}$$

Proof. An arbitrary euclidean vector (x, y) can be expressed in the base $\{\mathbf{V_1}, \mathbf{V_2}\}$ as

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{xy_2 - x_2y}{\delta}\mathbf{V_1} + \frac{x_1y - xy_1}{\delta}\mathbf{V_2} \tag{3-9}$$

$k\mathbf{V_3}$ is congruent to the null vector to modulus $\{\mathbf{V_1}, \mathbf{V_2}\}$ if it can be written as a linear combination of $\mathbf{V_1}$ and $\mathbf{V_2}$ with integer coefficients. Hence $k\mathbf{V_3} \equiv 0 \bmod \{\mathbf{V_1}, \mathbf{V_2}\}$ iff both $\frac{k(x_3y_2 - x_2y_3)}{\delta}$ and $\frac{k(x_1y_3 - x_3y_1)}{\delta}$ are integers. This condition amounts to $k\gamma$ and $k\chi$ both being multiples of δ:

$$\begin{cases} \delta | k\gamma \\ \delta | k\chi \end{cases} \tag{3-10}$$

Simplifying by $\gcd(\gamma, \delta)$ and $\gcd(\chi, \delta)$, this equation becomes

$$\begin{cases} \dfrac{\delta}{\gcd(\gamma, \delta)} \Big| k\dfrac{\gamma}{\gcd(\gamma, \delta)} \\ \dfrac{\delta}{\gcd(\chi, \delta)} \Big| k\dfrac{\chi}{\gcd(\chi, \delta)} \end{cases} \tag{3-11}$$

$\frac{\delta}{\gcd(\gamma, \delta)}$ and $\frac{\gamma}{\gcd(\gamma, \delta)}$ being mutually prime as well as $\frac{\delta}{\gcd(\chi, \delta)}$ and $\frac{\chi}{\gcd(\chi, \delta)}$ we deduce that k must be a multiple of both $\frac{\delta}{\gcd(\gamma, \delta)}$ and $\frac{\delta}{\gcd(\chi, \delta)}$ which proves the lemma.■

The set $\mathfrak{p}$ can be constructed in $\nu(\mathfrak{p})$ steps, each one involving a mod $(\mathbf{V_1}, \mathbf{V_2})$ reduction. The components ρ, σ of the mod $\{\mathbf{V_1}, \mathbf{V_2}\}$ reduction of an arbitrary vector (x, y) can be expressed as:

$$\begin{pmatrix} \rho \\ \sigma \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} - \left[\frac{xy_2 - x_2y}{\delta}\right]\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} - \left[\frac{x_1y - xy_1}{\delta}\right]\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \tag{3-12}$$

Even if this formula is fine, we will look for a yet simpler and faster way of generating the set $\mathfrak{p}$. In fact we have the following lemma, where δ, χ, γ are as above.

Lemma 3-4. *There is an integer* 2×2 *matrix* R *which maps the lattice* $L(\mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3})$ *bijectively onto* $L((\delta, 0), (0, \delta), (\chi, \gamma))$.

Proof. Using the classical identity for euclidean division

$$a = b\left[\frac{a}{b}\right] + \left\{\frac{a}{b}\right\} \tag{3-13}$$

the preceding identity can be simplified as follows:

$$\begin{pmatrix}\rho\\ \sigma\end{pmatrix} = \frac{1}{\delta}\left(\left\{\frac{xy_2 - x_2y}{\delta}\right\}\begin{pmatrix}x_1\\ y_1\end{pmatrix} + \left\{\frac{x_1y - xy_1}{\delta}\right\}\begin{pmatrix}x_2\\ y_2\end{pmatrix}\right) \tag{3-14}$$

But this can be written in matrix notation as

$$\begin{pmatrix}\rho\\ \sigma\end{pmatrix} = \frac{1}{\delta}\begin{pmatrix}x_1 & x_2\\ y_1 & y_2\end{pmatrix}\begin{pmatrix}\left\{\frac{xy_2 - x_2y}{\delta}\right\}\\ \left\{\frac{x_1y - xy_1}{\delta}\right\}\end{pmatrix} \tag{3-15}$$

revealing the rational unimodular matrix

$$U = \frac{1}{\delta}\begin{pmatrix}x_1 & x_2\\ y_1 & y_2\end{pmatrix} \tag{3-16}$$

We can then transform the situation and map the lattice $L(\mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3})$ to another lattice with the help of the matrix $R = \delta U^{-1} = \begin{pmatrix}y_2 & -x_2\\ -y_1 & x_1\end{pmatrix}$.

In the case where $x = x_3$ and $y = y_3$ the numbers $\left\{\frac{xy_2 - x_2y}{\delta}\right\}$ and $\left\{\frac{x_1y - xy_1}{\delta}\right\}$ become respectively the values χ and γ already introduced. Operator R maps $L(\mathbf{V_1}, \mathbf{V_2})$ bijectively to the subgroup $\{(m, n)\delta \mid m, n \in \mathbb{Z}\}$ and the lattice $L(\mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3})$ to the lattice generated by $(\delta, 0), (0, \delta), (\chi, \gamma)$, which is the assertion of Lemma 3-4. ■

The lattice $RL(\mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3})$ is doubly periodic of periods δ and δ. Through R the set $\mathfrak{p}$ is mapped to the modular sequence

$$(k\chi \bmod \delta, k\gamma \bmod \delta) \tag{3-17}$$

which can be written also as

$$\left(\left[\frac{k\chi}{\delta}\right], \left[\frac{k\gamma}{\delta}\right]\right), \qquad k = 0, 1, 2, \ldots, (\mathfrak{p}) \tag{3-18}$$

Thus the complexity in generating the set $\mathfrak{p}$ is reduced to the computation of two modular sequences, which can be done with additions and comparisons only, avoiding divisions. This can even be reduced once more by introducing the particular value of k, say η, for which

$$\eta\chi \equiv \gcd(\chi, \delta) \bmod \delta \tag{3-19}$$

For this value of k, the other sequence is equal to $\eta\gamma \bmod \delta$, that we shall denote ε. Obviously the set of points $R\mathfrak{p}$ can be built by the sequence

$$\left(k\gcd(\chi, \delta), \left\{\frac{k\varepsilon}{\delta}\right\}\right), \qquad k = 0, 1, 2, \ldots \nu(\mathfrak{p}) \tag{3-20}$$

which now needs only one modular computation for each step. Finally, with the former notations, we obtain:

Theorem 3-1. *The set* $\mathfrak{p}$ *can be built in* $\operatorname{lcm}\left(\frac{\delta}{\gcd(\chi, \delta)}, \frac{\delta}{\gcd(\gamma, \delta)}\right)$ *computations of a modular arithmetical sequence of type* $\left[\frac{k\varepsilon}{\delta}\right]$

3.3. Defining 3D digital lines

The lattice $\mathcal{L}$ introduced in section 3.2 is a rational lattice contained in the plane (P) $ax + by + cz = 0$. It is generated by the three rational vectors $\mathbf{V_1} = \pi(1, 0, 0)$, $\mathbf{V_2} = \pi(0, 1, 0)$ and $\mathbf{V_3} = \pi(0, 0, 1)$, where π is the orthogonal projection on (P).

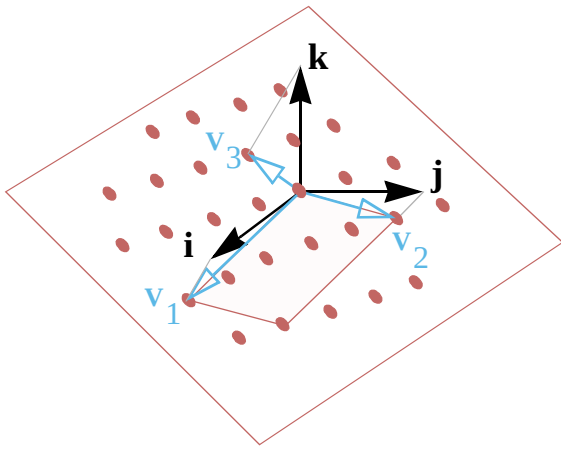


Figure 3-4: The lattice $\mathcal{L}$ as projection of the fundamental basis

The figure shows the fundamental basis $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ of $\mathbb{Z}^3$, its projection $(\pi(\mathbf{i}), \pi(\mathbf{j}), \pi(\mathbf{k}))$ on a plane (P) and the lattice $\mathcal{L}$ (brown dots).

These vectors can be easily written in terms of a, b, c and $\omega^2 = a^2 + b^2 + c^2$:

$$\mathbf{V_1} = \frac{1}{\omega^2}\begin{pmatrix} b^2+c^2 \\ -ab \\ -ac \end{pmatrix} \qquad \mathbf{V_2} = \frac{1}{\omega^2}\begin{pmatrix} -ab \\ a^2+c^2 \\ -bc \end{pmatrix} \qquad \mathbf{V_3} = \frac{1}{\omega^2}\begin{pmatrix} -ac \\ -bc \\ a^2+b^2 \end{pmatrix} \tag{3-21}$$

As these vectors are coplanar, we are in a situation almost similar to that of Section 3.2.1. To reduce it exactly to this case, we just have to clear out the denominator ω^2. The lattice generated by $\omega^2\mathbf{V_1}$ and $\omega^2\mathbf{V_2}$ is a rank two group isomorphic to $L(\mathbf{V_1}, \mathbf{V_2})$ of section 3.2.1. The only difference between both situations is that vectors $\omega^2\mathbf{V_1}$, $\omega^2\mathbf{V_2}$, $\omega^2\mathbf{V_3}$ now belong to $\mathbb{Z}^3$ instead of $\mathbb{Z}^2$, but all the preceding results go through, with the obvious modifications.

Theorem 3-2. *There is a 3×3 rational matrix R, which maps the lattice $\mathcal{L}$ bijectively on the sub-lattice $L((c,0),(0,c),(a,b))$ of $\mathbb{Z}^2$. This operator maps π fibers on lines which project on direction (a,b).*

This image $R\mathcal{L}$ is called the simplification (or reduction) of $\mathcal{L}$ and it is denoted by $\hat{\mathcal{L}}$.

In the same way image $R\mathfrak{p}$ is denoted $\hat{\mathfrak{p}}$ and image $RPar$ is denoted by $\hat{Par}$. Of course $\hat{Par} = [0,c)\times[0,c)$ is the new tile of the simplified lattice.

Proof. The projection π is not *invertible*, but we can still find operators which are almost inverses of it. (The map π being a fibration, such inverses are usually called *sections* of π). A possible section is given by the mapping

$$\begin{cases} \mathbf{V_1} \to (1, 0, 0) \\ \mathbf{V_2} \to (0, 1, 0) \\ (0, 0, 1) \to (0, 0, 1) \end{cases} \tag{3-22}$$

As π is the operator defined by

$$\begin{cases} (1, 0, 0) \to \mathbf{V_1} \\ (0, 1, 0) \to \mathbf{V_2} \\ (0, 0, 1) \to \mathbf{V_3} \end{cases} \tag{3-23}$$

its matrix (still denoted π) is

$$\pi = \frac{1}{a^2 + b^2 + c^2} \begin{pmatrix} b^2 + c^2 & -ab & -ac \\ -ab & a^2 + c^2 & -bc \\ -ac & -bc & a^2 + b^2 \end{pmatrix} \tag{3-24}$$

Thus our problem is to find the inverse of the matrix τ:

$$\tau = \frac{1}{a^2 + b^2 + c^2} \begin{pmatrix} b^2 + c^2 & -ab & 0 \\ -ab & a^2 + c^2 & 0 \\ -ac & -bc & a^2 + b^2 + c^2 \end{pmatrix} \tag{3-25}$$

A simple computation gives

$$\tau^{-1} = \frac{1}{c^2} \begin{pmatrix} a^2 + c^2 & ab & 0 \\ ab & b^2 + c^2 & 0 \\ ac & bc & c^2 \end{pmatrix} \tag{3-26}$$

If we let $R = c\tau^{-1}$ and evaluate the images of $\mathbf{V_1}$, $\mathbf{V_2}$ and $\mathbf{V_3}$ by R we respectively find the vectors

$$(c, 0, 0), (0, c, 0), (-a, -b, 0) \tag{3-27}$$

This proves that the operator R bijectively maps $L(\mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3})$ on the *integer* lattice $L((c, 0, 0), (0, c, 0), (-a, -b, 0))$. We also remark that this last one is the same as the lattice $L((c, 0, 0), (0, c, 0), (a, b, 0))$. ■

By definition (see Section 3.2), points of $\mathfrak{p}$ are the projections of π fibers. In this sense the image $\hat{\mathcal{L}}$ can be seen as the feet of all these fibers. But this strict planar interpretation is not the only one which can be deduced from the preceding computations. The most striking is the result of the computation of $R\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \frac{\omega^2}{c}\begin{pmatrix} a \\ b \\ c \end{pmatrix}$, because it proves that the lines directed by (a, b) are actually the projections of the *images of the fibers* by operator R, on the xOy plane. Besides, as $V_3 = \pi(0, 0, 1)$, the points of the form $k \cdot RV_3 = k(a, b)$, where $k \in \mathbb{Z}$, represent the sections of these fibers by the horizontal planes $z = k$.

Moreover the set $\mathfrak{p}$ of points of $L(\mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3})$ contained in the parallelogram built on $\mathbf{V_1}$ and $\mathbf{V_2}$ is mapped, by R on the set $\hat{\mathfrak{p}}$ of points of the 2D lattice $L((c, 0, 0), (0, c, 0), (-a, -b, 0))$ contained in the square $[0, c) \times [0, c) = \hat{Par}$ which is much simpler to study.

Of course, since $\mathbf{V_1}$ and $\mathbf{V_2}$, the respective projections of $(1, 0, 0)$ and $(0, 1, 0)$, are mapped onto $(c, 0)$ and $(0, c)$, the unit cubes of $\mathbb{Z}^3$ can be seen as the tiling induced by $\hat{Par} = [0, c) \times [0, c)$.

Finally we obtain the following:

The simplified lattice $\hat{\mathcal{L}}$ gives the intersection scheme of the euclidean line, directed by (a, b, c), with all the voxels of space.

A closer look at Figure 3-5 reveals this nice interpretation. The medium-sized and bigger points in Figure 3-5 represent the projection lattice of $\mathbb{Z}^3$ along the direction $(5, 9, 17)$. On this figure, the line that is drawn is the image of the fiber of the projection that goes through the origin. The points of the form $k(5, 9)$ on this line are the sections of the fiber by the horizontal planes $z = k$. The horizontal lines lc and the vertical lines mc are the respective images of the vertical planes $y = l$ and $x = m$. Thus by considering the positions of the points $k(a, b)$ for

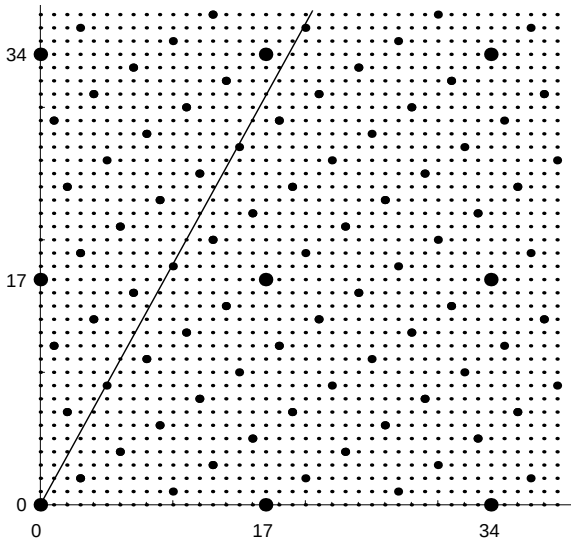


Figure 3-5: The reduction of the lattice $\mathcal{L}$ associated to $a = 5, b = 9, c = 17$

The smallest points are integer points of $\mathbb{Z}^2$, the medium ones are the points of $\hat{\mathcal{L}}$, while the largest ones belong to the lattice $L((c, 0), (0, c))$. The set $\hat{\mathfrak{p}}$ is made of the medium points contained in the square $\hat{Par} = [0, 17) \times [0, 17)$. The points located on the line of slope $\frac{9}{5}$, are the first multiples of the third vector $(a, b) = (5, 9)$. Their reduction modulo $c = 17$ give some of the points of $[0, 17) \times [0, 17)$.

$k = 0, 1, 2, \ldots$ we can say that the euclidean line directed by $(5, 9, 17)$ goes through the origin, then cuts the plane $z = 1$ in the square $[0, 1] \times [0, 1]$, the planes $z = 2$ and $z = 3$ in the square $[0, 1] \times [1, 2]$, the plane $z = 4$ in the square $[1, 2] \times [2, 3]$, ...

From this interpretation of the projection lattice we can derive a first notion of a 3D digital line:

Definition 3-1. The naive digital 3D line through the origin, directed by (a, b, c), (where $0 \le a < b < c$, $\gcd(a, b, c) = 1$, is given by the parametrization

$$\begin{cases} x = \left\lceil \frac{az}{c} \right\rceil \\ y = \left\lceil \frac{bz}{c} \right\rceil \end{cases} \tag{3-28}$$

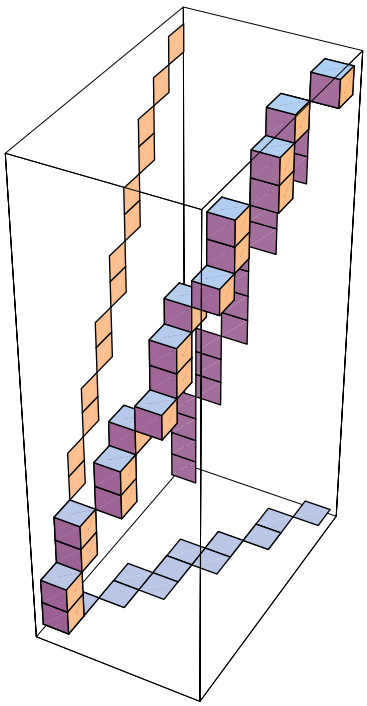


Figure 3-6: The digital line $D(5, 9, 17)$

or equivalently

$$\begin{cases} 0 \leq -cx + az < c \\ 0 \leq -cy + bz < c \end{cases} \tag{3-29}$$

It is denoted by $D(a, b, c)$.

We remark immediately that this notion is identical with the usual 3D discrete line built by the double 2D Bresenham algorithm [32] and whose arithmetic formulation based on the definition of 2D digital lines (Section 2-1) was given by Debled-Rennesson [15]. The set made by the feet of the fibers forming $D(a, b, c)$ is exactly $\mathfrak{p}$.

When considered in $\mathbb{R}^3$ instead of $\mathbb{Z}^3$, Equation 3-29 defines a generalized continuous cylinder having for intersection with the plane *xOy* (main plane orthogonal to the axis *Oz* along which the direction of the line has its biggest coordinate) a square of side 1. This provides another characterization of a digital 3D line:

Proposition 3-1. *The set of integer points contained in a continuous cylinder of axis* (a, b, c) *with* $0 \leq a < b < c$ and *whose intersection with the main plane* xOy *is a unit square, is a digital 3D line.*

3.4. Reading the topology of a 3D line on the projection lattice

From the previous interpretation of the reduced projection lattice $\hat{\mathcal{L}}$, we can see that the only steps between two consecutive points of the fiber that generate an x increment are those that cross one of the vertical lines mc, i.e., those taken from points close enough to a a vertical line mc. When reduced to modulo $\hat{Par}$ these points correspond to the points in a vertical strip $[c-a, c) \times [0, c)$. Similarly the only steps that generate a y increment are taken from points an horizontal strip $[0, c) \times [c-b, c)$ and the only points that generate both x and y increments at the same time are those take from points in the common region $[c-a, c) \times [c-b, c)$. Thus the fundamental square $\hat{Par} = [0, c) \times [0, c)$ of lattice $\hat{\mathcal{K}} = L((c, 0), (0, c))$ can be divided into four zones that we will denote as (1), (2), (3) and (4), see Figure 3-7. This gives a partition of $\hat{\mathfrak{p}}$ which governs the topology of the line $\mathcal{D}(a, b, c)$.

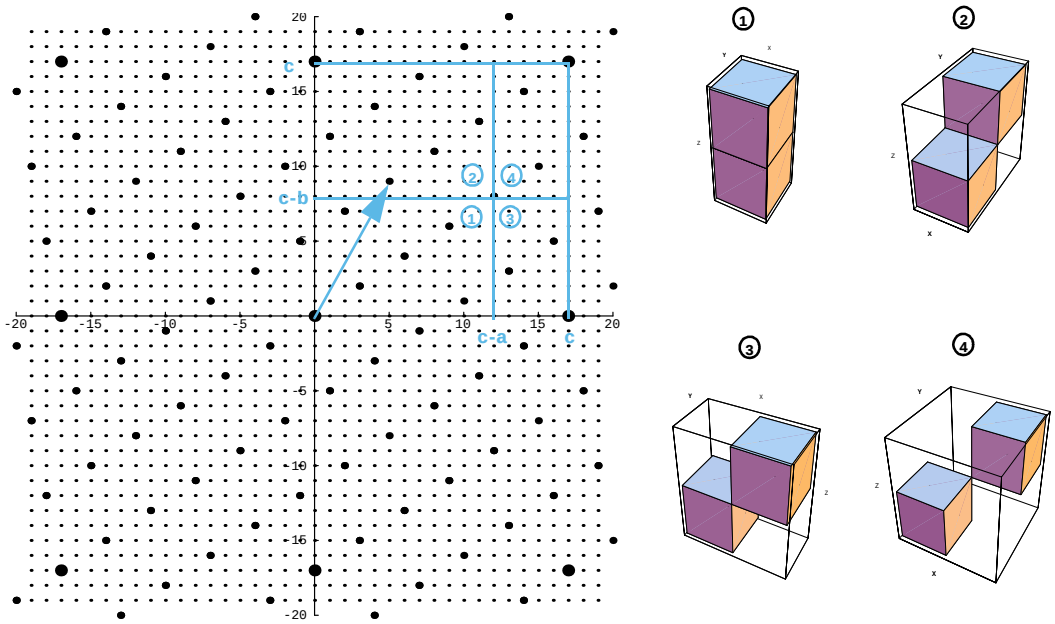


Figure 3-7: The four zones of the lattice associated with $D(5, 9, 17)$

We can see that along a naive 3D digital line of direction (a, b, c) as previously defined there are three types of adjacency only:

- strict *6-adjacency* when two voxels share one common face. 6-adjacency occurs along the z-axis only, i.e. shared faces are always parallel to the xOy coordinate plane. (Zone (1))
- strict *18-adjacency* or *edge-adjacency* when two voxels share one common edge. This type of adjacency occurs along the y-axis, (when the shared edge is parallel to the x-

axis) or along the x-axis (when the shared edge is parallel to the y-axis). (Zones (2) and (3)).

- strict *26-adjacency* or *corner-adjacency* when two voxels share one common vertex. This can occur for two of the eight vertices of a voxel only: the closest and the furthest from the origin. (Zone (4)).

We immediately deduce the following results from our description of 3D digital lines.

Proposition 3-2. *The number of face-adjacencies in one period of a naive 3D digital line of direction* (a, b, c) *is equal to the number of points of* $\hat{\mathcal{L}}$ *contained in the rectangle* $[0, c-a) \times [0, c-b)$ *(zone (1)).*

Proposition 3-3. *The number of edge-adjacencies along the* y*-axis in one period of a naive 3D digital line of direction* (a, b, c) *is equal to the number of points of* $\hat{\mathcal{L}}$ *contained in the rectangle* $[0, c) \times [c-b, c)$ *(zone (2)).*

Proposition 3-4. *The number of edge-adjacencies along the* x*-axis in one period of a naive 3D digital line of direction* (a, b, c) *is equal to the number of points of* $\hat{\mathcal{L}}$ *contained in the rectangle* $[c-a, c) \times [0, c)$ *(zone (3)).*

Proposition 3-5. *The number of corner-adjacencies in one period of a naive 3D digital line of direction* (a, b, c) *is equal to the number of points of* $\hat{\mathcal{L}}$ *contained in the rectangle* $[c-a, c) \times [c-b, c)$ *(zone (4)).*

Theorem 3-3. *The three projections onto the main planes (xOy) (yOz) and (xOz) of the naive 3D digital line* $D(a, b, c)$ *are naive 2D digital lines iff* $\hat{\mathfrak{p}} \cap [c-a, c) \times [0, c-b) = \varnothing$

3.5. Combinatorially distinct 3D lines

Definition 3-1 shows that our approach contains the former classical digital lines, but also many others that we can build by an extension of the previous notion. This situation is similar to that of 2D lines [46]. Up to this point we have defined and built 3D digital lines from the points of $\hat{\mathcal{L}}$ contained in one of the fundamental domains of $\hat{\mathcal{K}} = L((c, 0), (0, c))$. In fact we can also consider the collections of fibers whose feet are contained in other fundamental domains B of $\hat{\mathcal{K}}$. We can prove that each time $B \cap \mathcal{L}$ is *8-connected* for the topology of the *minimal basis* of $\hat{\mathcal{L}}$ then $\pi^{-1}(B)$ is a valuable notion of a 3D digital line.

We can extend the idea even further and consider domains over $\hat{\mathcal{L}}$ other than fundamental domains of $\hat{\mathcal{K}}$. An especially interesting case consists of fundamental domains of lattices

$\hat{\mathcal{T}}(s,t)$ generated from integer affine translations of $\hat{\mathcal{K}}$. As the fundamental domain $\hat{Par}$ of $\hat{\mathcal{K}}$ contains c^2 integer points, there exists c^2 such lattices $\hat{\mathcal{T}}(s,t)$, $(s,t) \in [0,c) \times [0,c)$. For each possible integer translation of vector (s,t) of $\hat{Par}$ it is possible to build a new 3D digital line leading to c^2 different digital lines of direction (a,b,c). Actually these c^2 different lines can be grouped into c classes of c digital lines having an equivalent structure within integer 3D translation. Thus given an integer direction (a,b,c) there exist c combinatorially distinct possible structures of the corresponding digital line. This is due to the fact that any fundamental domain of $\hat{\mathcal{L}}$ has an integer area of c and tiles $\hat{Par}$ into c subtiles (Figure 3-8 and Figure 3-9).

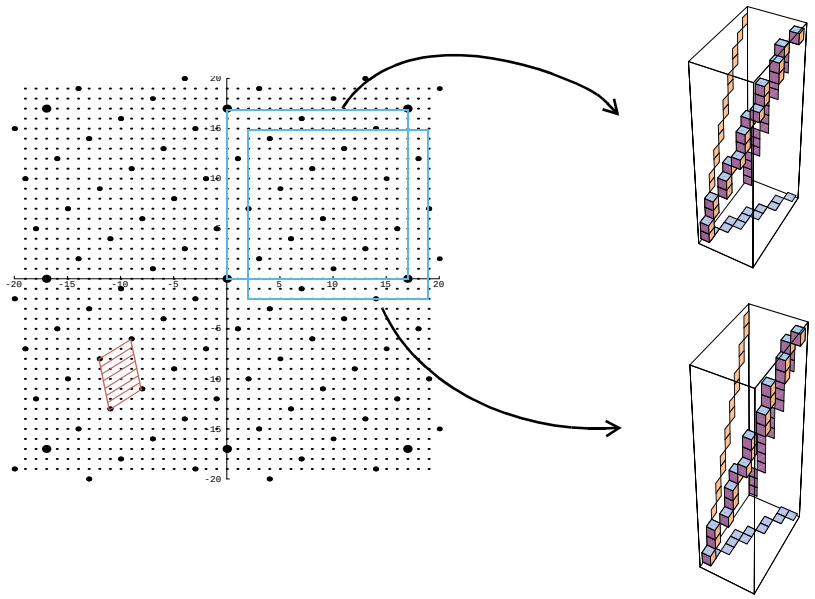


Figure 3-8: Reading the structure of combinatorially distinct lines from $\mathcal{L}$

Two affine translations of the fundamental domain of $\hat{\mathcal{K}}$ and the corresponding 3D digital lines. The hatched area represents the fundamental domain of $\hat{\mathcal{L}}$ which has an area of c.

This relation between the simplified lattice $\hat{\mathcal{L}}$ and the topological structure of 3D digital lines offers a new interesting point of view. It could certainly lead to a method for determining which line among the combinatorial variations is the closest digital connected set to a n euclidean line though this remains to be done.

3.6. Intersecting discrete 3D lines and planes

We now apply Definition 3-1 to a particular problem: determining the intersection of a digital 3D line and a digital plane. Section 2.5.1 has shown that the intersection of digital sets can be particularly complex even though the intersection of the euclidean counterpart is very simple. Intersecting a line and a plane is a very basic operation in computer graphics and is used in all

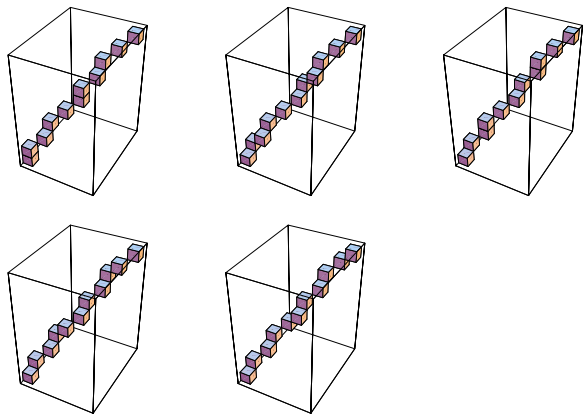


Figure 3-9: Combinatorial variations of $D(3, 4, 5)$

The 5 combinatorially distinct possible structures of the digital line of direction (3,4,5). Two periods are represented in each example.

ray-casting algorithms. Though the vast majority of ray-tracing implementations involves continuous models and thus euclidean lines and planes, some attempts at a full discrete version of the ray tracing algorithm have been made especially by Yagel, Cohen an Kaufman [57]. In such an algorithm intersecting a discrete ray (3D digital line) with the surface of an object (which may be represented by a surface mesh, i.e. pieces of digital planes assembled together) is the core operation. As such, it is the most consuming part and most of the effort is generally put in accelerating that calculation sometimes neglecting some theoretical aspects. The following study on the other hand, tries to emphasize the precision of the discrete intersection calculation from a theoretical point of view and describes the exact set of discrete points that make up the intersection. This could find for instance a possible application in the anti-aliasing of ray-traced images following ideas developed by Amanatides [1].

Let us consider the naive digital line $D(a, b, c, \mu, \mu')$ with a definition generalizing the one given in Definition 3-1:

$$D(a, b, c, \mu, \mu') : \quad \begin{cases} \mu \le -cx + az < \mu + c \\ \mu' \le -cy + bz < \mu' + c \end{cases} \tag{3-30}$$

where all parameters are integers and the direction (a, b, c) verifies the usual restrictions, i.e. a, b and c are mutually prime and strictly positive (see Section 2.1). As already mentioned a digital line of direction (a, b, c) defined in this way can be seen as the intersection of two digital planes orthogonal to the plane of normal (a, b, c) and respectively to the (xOz) and (yOz) main coordinate planes. Therefore, the intersection of a naive 3D digital line with a digital plane is in fact the intersection of three digital planes which is similar to the equivalent continuous problem.

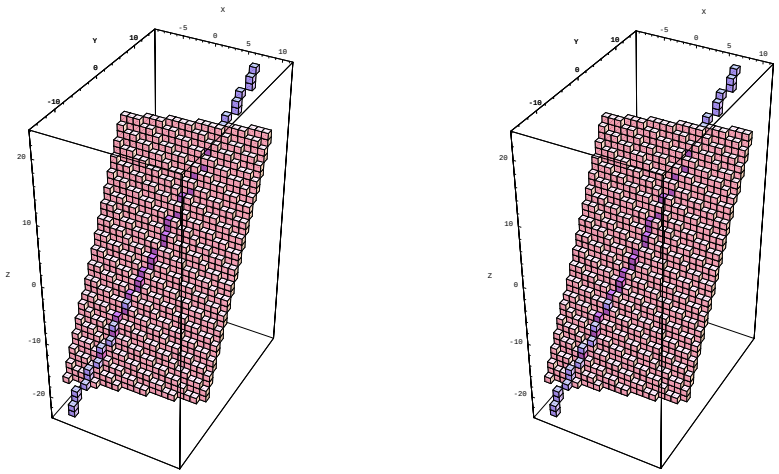


Figure 3-10: Intersection of $D(7, 15, 23, 0)$ **and** $P(-8, 29, -15, -25)$

We want to determine the intersection of $D(a, b, c, \mu, \mu')$ with the digital plane $P(d, e, f, \gamma, \rho)$ defined as in Definition 2-2 with the additional restriction $ad + be + cf > 0$. The quantity being non null avoids the degenerated cases where the line and plane have no intersection or the line is contained in the plane. The expression being positive can always be guaranteed since considering one of the opposite vectors $-(a, b, c)$ or $-(d, e, f)$ does not modify the intersection.

The following matricial system defines this intersection:

$$\begin{pmatrix} \mu \\ \mu' \\ \gamma \end{pmatrix} \leq \begin{pmatrix} -c & 0 & a \\ 0 & -c & b \\ d & e & f \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} < \begin{pmatrix} \mu + c \\ \mu' + c \\ \gamma + \rho \end{pmatrix} \tag{3-31}$$

Of course this system must be solved in $\mathbb{Z}^3$. The key idea is to find an appropriate unimodular matrix $\mathcal{U}$ (i.e. having a determinant equal to 1) that makes the system easier to solve. The important fact concerning $\mathcal{U}$ is that it is a bijective transform of $\mathbb{Z}^3$. In other words we want to map the intersection points from the original space where coordinates are denoted (x, y, z) into another one where they are denoted (X, Y, Z) and where the set of points of the intersection appears more regular.

Let (u, v) and (u', v') two pairs of integers such that

$$au + cv = 1 \tag{3-32}$$

and

$$bu' + cv' = 1 \tag{3-33}$$

We find successively:

$$\underbrace{\begin{pmatrix} -c & 0 & a \\ 0 & -c & b \\ d & e & f \end{pmatrix}\begin{pmatrix} -v & 0 & a \\ 0 & -1 & 0 \\ u & 0 & c \end{pmatrix}}_{\mathcal{U}_1} = \begin{pmatrix} 1 & 0 & 0 \\ bu & c & bc \\ -dv + fu & -e & ad + cf \end{pmatrix} \tag{3-34}$$

We introduce the integers d', e' and f' to simplify the notations:

$$\begin{cases} d' = -dv + fu \\ e' = -e \\ f' = ad + be + cf \end{cases} \tag{3-35}$$

we have then

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 \\ bu & c & bc \\ -dv + fu & -e & ad + cf \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -b \\ 0 & 0 & 1 \end{pmatrix}}_{\mathcal{U}_2} = \begin{pmatrix} 1 & 0 & 0 \\ bu & c & 0 \\ d' & e' & f' \end{pmatrix} \tag{3-36}$$

In addition we require that e' and f' be relatively prime which ensures that there exist (m, n) such as:

$$me' + nf' = 1 \tag{3-37}$$

Then we can write

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 \\ bu & c & 0 \\ d' & e' & f' \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ -md' & 1 & 0 \\ -nd' & 0 & 1 \end{pmatrix}}_{\mathcal{U}_3} = \begin{pmatrix} 1 & 0 & 0 \\ bu - md'c & c & 0 \\ 0 & e' & f' \end{pmatrix} \tag{3-38}$$

and finally we get $\mathcal{U}$

$$\mathcal{U} = \mathcal{U}_1\mathcal{U}_2\mathcal{U}_3 = \begin{pmatrix} -v - and' & 0 & a \\ d'm - bnd' & -1 & b \\ u - cnd' & 0 & c \end{pmatrix} \tag{3-39}$$

$\mathcal{U}$ transforms Equation 3-31 into

$$\begin{pmatrix} \mu \\ \mu' \\ \gamma \end{pmatrix} \leq \begin{pmatrix} 1 & 0 & 0 \\ bu - md'c & c & 0 \\ 0 & e' & f' \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} < \begin{pmatrix} \mu + c \\ \mu' + c \\ \gamma + \rho \end{pmatrix} \tag{3-40}$$

with

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathcal{U} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{3-41}$$

Since the scalar product $f' = ad + be + cf$ is strictly positive according to our initial hypotheses, we can rewrite Equation 3-40 as the following expression:

$$\begin{cases} \mu \leq X < \mu + c \\ Y = md'X - \left\lfloor \dfrac{buX - \mu}{c} \right\rfloor \\ -\left\lfloor \dfrac{e'Y - \gamma}{f'} \right\rfloor \leq Z < -\left\lfloor \dfrac{e'Y - \gamma - \rho}{f'} \right\rfloor \end{cases} \tag{3-42}$$

from which we can immediately deduce the following intersection scanning algorithm:

where δ denotes the quantity $\left\{ \dfrac{e'Y - \gamma}{f'} \right\}$.

Equation 3-42 shows that

- X steps through c values
- for each of X there is a unique value of Y
- for each value of Y there are:

```
for X = μ to X = μ + c − 1 do {
    Y = md′X − ⌈(buX − μ)/c⌉
    if (ρ − δ) > 0 then do
        for Z = −⌈(e′Y − γ)/f′⌉ to Z = −⌈(e′Y − γ − ρ)/f′⌉ do
            PlotVoxel(u · (X, Y, Z))
}
```

Figure 3-11: A digital line/plane intersection algorithm

- $-\left\lceil \frac{-\rho}{f'} \right\rceil$ values for Z if $\delta < f' - \left\{ \frac{-\rho}{f'} \right\}$
- $-\left\lceil \frac{-\rho}{f'} \right\rceil - 1$ values for Z if $\delta \geq f' - \left\{ \frac{-\rho}{f'} \right\}$

Proposition 3-6 follows immediately:

Proposition 3-6. *The intersection of a naive three-dimensional line* $D(a, b, c, \mu)$ *and a digital plane* $P(d, e, f, \gamma, \rho)$ *consists of* N *voxels with*

$$-c\left\lceil \frac{-\rho}{f'} \right\rceil - c \leq N \leq -c\left\lceil \frac{-\rho}{f'} \right\rceil \tag{3-43}$$

The complexity of the algorithm to calculate the intersection between a 3D digital line and a digital plane is $O(N)$ where N is the number of voxels contained in the intersection.

3.7. Incremental intersection of parallel adjacent 3D lines with a plane

For some common applications like parallel projections it may be necessary to compute the intersection of a digital plane with a set of parallel adjacent digital lines. The classical approach trying to calculate these intersections incrementally based on euclidean geometry and floating point arithmetics is sensitive to error accumulation and may be tedious to adjust. On the other hand, the simple algorithm presented previously has very interesting properties which offer a particularly efficient solution to the problem and avoids numerical drifts.

We shall carry the demonstration for a set of naive parallel digital lines adjacent along the x-axis. The problem being perfectly symmetrical, the results will be easily transposed to adjacency along the two other axes.

According to Equation 3-30, we define the naive 3D digital line of direction (a, b, c) going through the integer point (x_0, y_0, z_0) as the subset of $\mathbb{Z}^3$ verifying:

$$\begin{cases} -cx_0 + az_0 \leq -cx & + az < -cx_0 + az_0 + c \\ -cy_0 + bz_0 \leq & -cy + bz < -cy_0 + bz_0 + c \end{cases} \tag{3-44}$$

Assuming that the intersection between the line going through (x_0, y_0, z_0) and the plane $P(d, e, f, \gamma, \rho)$ is known, we want to determine the intersection of this plane with the line going through $(x_0 + 1, y_0, z_0)$.

We denote with the $_{x0}$ subscript, values related to the intersection with the line going through (x_0, y_0, z_0) and with the $_{x0+1}$ subscript, values related to the intersection with the line going through $(x_0 + 1, y_0, z_0)$. We also denote with η_{x_0} the following quantity $\eta_{x_0} = \left\lfloor \dfrac{\delta_{x_0} + \left\{ \dfrac{d}{f'} \right\}}{f'} \right\rfloor$ which can assume only two values: 0 or 1.

A simple calculation making use Equations 3-39 to 3-41 yields:

$$\begin{aligned} X_{x_0+1} &= X_{x_0} - c \\ Y_{x_0+1} &= Y_{x_0} + bu - md'c \\ Z_{inf_{x_0+1}} &= Z_{inf_{x_0}} + u - nd'c - \left\lfloor \frac{d}{f'} \right\rfloor - \eta_{x_0} \\ \delta_{x_0+1} &= \delta_{x_0} + \left\{ \frac{d}{f'} \right\} - f'\eta_{x_0} = \left\{ \frac{\delta_{x_0} + \left\{ \dfrac{d}{f'} \right\}}{f'} \right\} \end{aligned} \tag{3-45}$$

where Z_{inf} represents the lower bound of the interval of possible values of Z for a given Y (Equation 3-42).

These results can be translated into the (x, y, z) space where they appear simpler:

$$\begin{aligned} x_{x_0+1} &= x_{x_0} + 1 - a\left\lfloor \frac{d}{f'} \right\rfloor - a\eta_{x_0} \\ y_{x_0+1} &= y_{x_0} - b\left\lfloor \frac{d}{f'} \right\rfloor - b\eta_{x_0} \\ z_{x_0+1} &= z_{x_0} - c\left\lfloor \frac{d}{f'} \right\rfloor - c\eta_{x_0} \end{aligned} \tag{3-46}$$

Then we derive immediately from these equations the algorithm in Figure 3-12 and Figure 3-13 which is presented without full optimization for the sake of clarity. The calculation

```
Definitions and allocations
Define type Voxel as a record of 3 integers denoted x, y, z
nbInterMax=-div(ρ,f')                    The maximum number of intersection point for each
                                         X value

Allocate interVox as a 2D Array of Voxel of size (c,nbInterMax)
Allocate δ as an Array of Integer of size c
Allocate N as an Integer                 The number of intersection points for a given X

Initialization for the first line
For i from 0 to c-1 do
   X = a*z0 - c*x0 + i
   Y = m*d'*X - div(b*u*X,c)
   Z = -div(e'*Y - γ, f')
   δ(i) = mod(e'*Y - γ, f')
   If δ(i) + mod(-ρ,f') >= f' then
      N(i) = nbInterMax
   Otherwise
      N(i) = nbInterMax-1
   For j from 0 to nbInterMax-1 do
      interVox(i,j) = matrixVectorMultiply (U, Vector(X,Y,Z))
   For j from 0 to N-1 do
      PlotVoxel(interVox(i,j))
```

Figure 3-12: Incremental plane/multi-line intersection algorithm (initialization)

of consecutive intersections with adjacent lines involves only additions and a few tests. It is therefore very efficient.

3.8. Summary

The in-depth study of 3D digital lines is a largely open subject and few results are available. Since the developments in this area of research are largely driven by the necessities of practical applications, people generally focus on algorithmics more than on theoretical aspects. Even though the third dimension makes the problem much more intricate, there exist promising new

```
Loop for the remaining lines
For xOffset from 0 to numberOfLines-1 do
   For i from 0 to c-1 do
      If δ(i) + mod(d,f') >= f' then
         δ(i) += mod(d,f') - f'
         For j from 0 to nbInterMax-1 do
            interVox(i,j).x += - a*div(d,f') - a + 1
            interVox(i,j).y += - b*div(d,f') - b
            interVox(i,j).z += - c*div(d,f') - b
      Otherwise
         δ(i) += mod(d,f')
         For j from 0 to nbInterMax-1 do
            interVox(i,j).x += - a*div(d,f') - a + 1
            interVox(i,j).y += - b*div(d,f') - b
            interVox(i,j).z += - c*div(d,f') - b
      If δ(i) + mod(-ρ,f') >= f' then
         N = nbInterMax
      Otherwise
         N = nbInterMax-1
      For j from 0 to N-1 do
         PlotVoxel(interVox(i,j))
```

Figure 3-13: Incremental plane/multi-line intersection algorithm (continuation)

approaches. Thus, using the projection of $\mathbb{Z}^3$ onto an euclidean plane, we have shown that the structure of a 3D digital line is in correspondence with the structure of a 2D integer lattice. This approach considerably extends the classical notion based on 2D Bresenham lines and opens the doors to new results. Among these we have presented a classification of lines of a given direction into classes of equivalent combinatorial structure and suggested that the same approach could lead to a solution to the problem of the closest digital connected set to an euclidean line.

An algebraic definition, equivalent to the classical definition of 3D lines based 2D Bresenham lines, can be deduced from our approach. We used it to solve the intersection between a 3D digital line and a digital plane. Unlike usual algorithms based on euclidean geometry that content themselves with the integer point that is closer to the real intersection of the euclidean objects, we are able to determine precisely in $O(N)$ time (N being the number of voxels in the intersection) the exact intersection between the digital line and plane, no matter how complex that intersection is. Moreover, unlike algorithms based on continuous geometry which are very sensitive to error accumulation when extended to the incremental calculation of the intersections of a plane with a series of parallel lines, our result can be generalized with no numerical drift to solve the intersection of adjacent digital lines with the same digital plane in an incremental manner also in $O(N)$ time but with much simpler calculations.

4 Digitization of Bézier Curves and Surfaces

This chapter introduces a new subdivision criterion leading to a scan-conversion algorithm of Bézier curves and surface patches that is compatible with the results of discrete geometry and does not rely on arbitrary precision constants.

4.1. Introduction

Bézier curves and patches are among the most fundamental primitives in computer graphics and computer aided modeling. However, as they are defined by means of mathematical equations, they are continuous objects which are not ideally suited for a computer representation. Discrete geometry aims at providing an equivalent of these mathematical objects in the same way as it has established formal definitions of digital lines and planes [46]. A first step towards this goal consists in developing a discretization algorithm of continuous Bézier curves and patches which would be consistent with existing results in discrete geometry. In this section, we propose such an algorithm. Our approach is based on a classical De Casteljau recursive subdivision algorithm but with a new flatness criterion based on the digital geometry of lines and planes which guarantees a recursion depth close to optimal and appropriate geometric and topological characteristics of the obtained discrete curve or surface with no need for arbitrary constants. Moreover, though we restrict our presentation here to the case of cubic Bézier curves and patches for the sake of clarity, it is a remarkable fact that the approach is general enough to be easily extended to higher degrees and dimensions.

4.2. Existing approaches to the problem

There exist essentially two different approaches to the scan-conversion of Bézier curves and surface patches [23]. The first one uses the parametric representation of the curve and evaluates repetitively the equations using forward differencing. Forward differencing is a fast and efficient technique which can be hardware accelerated. However it suffers from two major drawbacks: first, it is a floating point algorithm subject to numerical drifts due to error accumulation [8] and whose implementation requires great care and a register width depending on the number of pixels to draw, furthermore there is naturally no linear relation between the parameter and the coordinates of the drawn points. Hence a regular subdivision of the parameter interval, though simple, is particularly inefficient since it can lead to many unneeded evaluations drawing the same discrete point (if the parameter interval is too small) or holes in the curve respectively patch (if the parameter interval is too big). Therefore refinements such as dynamic step size adjustment are preferred [8, 21, 55]. But even in that case there is still a need for choosing a parameter increment and criterions for deciding when to scale that increment [21]. Unfortunately all those criterions are based on some geometrical value (surface of a triangle, distance between a point a line, angle) being “small” and thus require determining a tolerance

constant which, in practice is often chosen arbitrarily and whose relation to the sampling grid is not clear.

The second approach uses the De Casteljau algorithm [19], a stable and efficient method with intrinsic adaptiveness to the curve. This method consists in recursively subdividing the control polygon $(P_0^0, P_1^0, P_2^0, P_3^0)$ into two sub-polygons $(P_0^0, P_0^1, P_0^2, P_0^3)$ and $(P_0^3, P_1^2, P_2^1, P_3^0)$ (Figure 4-1) where the P_j^i are defined as weighted sums of polygon vertices:

$$P_i^{n+1} = \alpha P_i^n + (1-\alpha) P_{i+1}^n \qquad \alpha \in [0, 1] \qquad (4\text{-}1)$$

At each step the area of a new control polygon is smaller and hence is closer to the Bézier arc which remains invariant along the process. The recursion can be stopped when the control polygon is close enough to the curve.

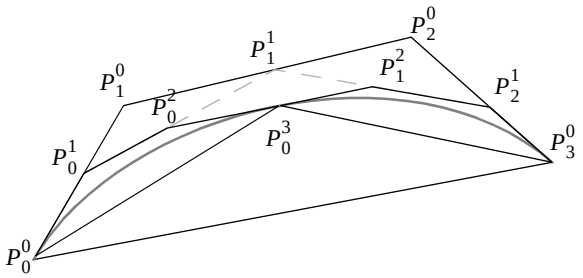


Figure 4-1: De Casteljau Subdivision

Theorems exist that indicate when the maximum distance between the arc and the control polygon is smaller than ε, based on the geometry of the sub-polygons [39, 56] or directly based on the initial control polygon and the recursion depth [34]. All of these results rely on the choice of an ad hoc constant ε which makes them quite unsatisfying from a theoretical point of view. In what follows we show that we can eliminate the need for such a constant.

4.3. Polygonalization of cubic Bézier curves using digital lines

We denote with $[r]$ the integer part of $r \in \mathbb{R}$, i.e., the greatest integer smaller than r. Similarly we denote with $[R]$ the integer point of $\mathbb{Z}^2$ (resp. $\mathbb{Z}^3$) whose coordinates are the respective integer parts of the coordinates of $R \in \mathbb{R}^2$ (resp. $\mathbb{R}^3$). Let us consider an arc of integer endpoints (P, Q). We call *axis* of the arc, the line defined by its two endpoints. The vector $\mathbf{PQ}$ is the *direction* of the arc. We also call *width* of the arc, the diameter of the smallest cylinder of axis (P, Q) that encompasses the whole arc. And similarly we call *width* of a set of points of $\mathbb{R}^2$,

$E = \{P_i\}_{0 \le i < n}$ with respect to the direction $(a, b) \in \mathbb{Z}^2$, the diameter of the narrowest cylinder of axis (a, b) that encompasses E. This width $w_{(a, b)}(E)$ is given by:

$$w_{(a, b)}(E) = \frac{dia(\{aP_{ix} + bP_{iy}\}_{0 \le i < n})}{\sqrt{a^2 + b^2}} \tag{4-2}$$

where *dia(A)* for a subset X of $\mathbb{R}$ is defined as:

$$dia(X) = max_{x \in X}(x) - min_{x \in X}(x) \tag{4-3}$$

Theorem 4-1. *Let C be a planar arc of integer endpoints P, Q and of width w. If* $w \le \frac{max(|\mathbf{PQ}_x|, |\mathbf{PQ}_y|)}{\|\mathbf{PQ}\|}$ *then the best 8-connected integer approximation of C is a naive digital straight line segment of direction* **PQ**.

Proof. Equation 2-4 defines a naive digital line as the set of integer points contained in a continuous strip of the euclidean plane. The width w of this euclidean strip relates to the arithmetic thickness of the digital line ρ through the relation $w = \frac{\rho}{\sqrt{a^2 + b^2}}$ (see Section 2.2). It becomes clear then, that if C verifies $w \le \frac{max(|\mathbf{PQ}_x|, |\mathbf{PQ}_y|)}{\|\mathbf{PQ}\|}$ then it fits within the real boundaries of a naive digital line and hence there cannot be a better integer approximation to C than a naive digital straight line segment. ■

Theorem 4-2. *Let B be a planar cubic Bézier arc defined by its control polygon* (P_0, P_1, P_2, P_3)*, B can be optimally represented by a naive digital straight line segment of direction* $[P_0P_3]$ *iff*

$$dia(\{aP_{ix} + bP_{iy}\}_{0 \le i \le 3}) < \frac{4}{3} max(|a|, |b|) \tag{4-4}$$

where $a = [P_{3y}] - [P_{0y}]$ *and* $b = -[P_{3x}] + [P_{0x}]$

Proof. Wang's theorem states that B lies at most $\frac{3}{4} max(d(P_1), d(P_2))$ away from its axis where $d(P_i)$ denotes the distance of point P_i to the axis and that this bound is optimal [56]. Furthermore the convex hull containment property of Bézier curves states that B lies entirely within its control polygon [19]. Therefore the width of B is at most 3/4 of the width of its

control polygon with respect to the line $([P_0], [P_3])$, the width of the control polygon being given by $\frac{dia(\{aP_{ix} + bP_{iy}\}_{0 \le i \le 3})}{\sqrt{a^2 + b^2}}$. The result then follows directly from Theorem 4-1. ■

A control polygon verifying Theorem 4-2 is said to be *flat* and the recursive subdivision can be stopped at that level. If the control polygon does not meet the criterion of Theorem 4-2 then it is subdivided into two sub-polygons according to Equation 4-1. Equation 4-1 leaves a degree of liberty in the choice of α. Optimizing this value at each recursion step yields a digitization with the minimum number of discrete segments. Such optimization however falls beyond the scope of this work and in practice $\alpha = \frac{1}{2}$ is the usual choice.

Theorem 4-2 does not provide the complete equation of the digital line segment that is the best approximation of the spline arc but only its direction. The affine offset γ must also be determined. The ideal value of γ which yields a discretization by the closest integer corresponds to the midline (axis) of the narrowest cylinder of direction $([P_0], [P_3])$ enclosing the curve within its control polygon. In order to find an algebraic formulation of γ, we must distinguish two cases depending on whether the curve crosses the line $([P_0], [P_3])$ or not.

1. The curve does not cross $([P_0], [P_3])$**.** In this case the curve lies between the line $([P_0], [P_3])$ and a parallel at three fourths of the distance of the most distant point of the control polygon (which may be P_1 or P_2) to that line. Assuming that point to be P_1, γ becomes:

$$\gamma = \left\lceil \frac{3(aP_{1x} + bP_{1y}) + 5(aP_{0x} + bP_{0y})}{8} \right\rceil - \left\lceil \frac{max(|a|, |b|)}{2} \right\rceil \qquad (4\text{-}5)$$

2. The curve crosses $([P_0], [P_3])$**.** In this case the curve is enclosed in a cylinder of direction $[\mathbf{P_0P_3}]$ whose axis is midway between the points P_1 and P_2. Hence γ becomes:

$$\gamma = \left\lceil \frac{(aP_{1x} + bP_{1y}) + (aP_{2x} + bP_{2y})}{2} \right\rceil - \left\lceil \frac{max(|a|, |b|)}{2} \right\rceil \qquad (4\text{-}6)$$

A criterion for digitizing 3D Bézier curves stems from the same principle as for 2D Bézier curves: the De Casteljau recursion stops and the curve segment can be rendered as a discrete line with no loss of precision when the convex hull of its control polygon is bounded by the limits of a 3D digital line as defined in Equation 3-29. Denoting $a = [P_{3x}] - [P_{0x}]$,

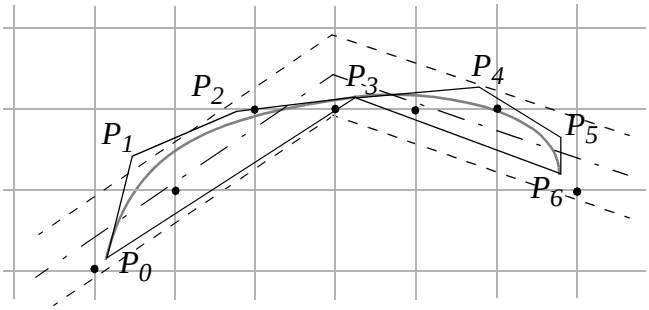


Figure 4-2: Discretization of a Bézier arc

A Bézier arc described by two flat control polygon (P_0,P_1,P_2,P_3) and (P_3,P_4,P_5,P_6). The round dots are the integer points making up the discretization of the Bézier arc. The dash-dotted lines represent the axes defined by Equation 4-5 while the simple dashed lines represent the real boundaries of the digital line segments as defined by Equation 2-4.

$b = [P_{3y}] - [P_{0y}]$, $c = [P_{3z}] - [P_{0z}]$ and assuming $|c| = max(|a|, |b|, |c|)$, the condition writes:

$$\begin{cases} dia(\{-cP_{ix} + aP_{iz}\}_{0 \le i \le 3}) \le \frac{4}{3}|c| \\ dia(\{-cP_{iy} + bP_{iz}\}_{0 \le i \le 3}) \le \frac{4}{3}|c| \end{cases} \quad (4\text{-}7)$$

The affine offset proposed in Equations 4-5 and 4-64-6 still holds, with the same restrictions, in this case by considering independently the projections of the control polygon on the main planes *xOz* and *yOz*.

4.4. Digitization of Bézier surface patches

We call *naive digital plane patch* a non-empty 26-connected subset of a naive digital plane $P(a, b, c, \gamma)$ defined as a non-empty polygon on that plane by the following set of equations:

$$Q(a, b, c, \gamma, (\mathbf{q}_k, \lambda_k)_{0 \le k \le m}): \quad P(a, b, c, \gamma) \cap \begin{cases} q_{0x}x + q_{0y}y + q_{0z}z < \lambda_0 \\ q_{1x}x + q_{1y}y + q_{1z}z < \lambda_1 \\ \dots \\ q_{(m-1)x}x + q_{(m-1)y}y + q_{(m-1)z}z < \lambda_3 \end{cases} \quad (4\text{-}8)$$

where $\mathbf{q}_k \in \mathbb{Z}^3$ and $\lambda_k \in \mathbb{Z}$ for $0 \le k \le m$.

Special interesting cases of naive digital plane patches include digital quadrilaterals ($m = 4$) and triangles ($m = 3$).

3D bicubic tensor-product Bézier patches as well as Bézier triangles can be approximated by naive digital plane patches as defined by Equation 4-8. Indeed the De Casteljau recursive construction is general and still applies in those cases.

Theorem 4-3. *Let B be a bicubic Bézier patch defined by its control net* $(P_{i,j})_{0 \le \substack{i \\ j} \le 3}$. *Let* $\mathbf{u} = [\mathbf{P_{0,0}}][\mathbf{P_{3,0}}]$, $\mathbf{v} = [\mathbf{P_{0,0}}][\mathbf{P_{3,3}}]$ *and* $\mathbf{n} = \mathbf{u} \times \mathbf{v} = (a, b, c)$. *If*

$$dia\left(\{a(P_{i,j})_x + b(P_{i,j})_y + c(P_{i,j})_z\}_{0 \le \substack{i \\ j} \le 3}\right) \le max(|a|, |b|, |c|) \qquad (4\text{-}9)$$

then the best 26-connected integer approximation of B is a digital plane patch of normal direction **n**.

Theorem 4-4. *Let B be a Bézier triangular patch of degree d defined by its control net* $(P_{ijk})_{\substack{0 \le i,j,k \le d \\ i+j+k = d}}$. *Let* $\mathbf{u} = [\mathbf{P_{003}}][\mathbf{P_{300}}]$, $\mathbf{v} = [\mathbf{P_{003}}][\mathbf{P_{030}}]$ *and* $\mathbf{n} = \mathbf{u} \times \mathbf{v} = (a, b, c)$. *If*

$$dia\left(\{a(P_{ijk})_x + b(P_{ijk})_y + c(P_{ijk})_z\}_{\substack{0 \le i,j,k \le d \\ i+j+k = d}}\right) \le max(|a|, |b|, |c|) \qquad (4\text{-}10)$$

then the best 26-connected integer approximation of B is a digital plane triangle of normal direction **n**.

Proof. Since the containment property of Bézier patches and triangles within their control net holds, Equations 4-9 and 4-10 being verified amounts to the whole surface *B* lying between the limits defined by Equation 2-8 for a naive digital plane of direction (a, b, c). ■

The value of the affine offset γ of the digital plane that represents the digitization of *B* by the closest integer point is given by:

$$\gamma = \left\lceil \frac{min(a(P_{i,j})_x + b(P_{i,j})_y + c(P_{i,j})_z)_{0 \le \substack{i \\ j} < 3}}{2} + \frac{max(a(P_{i,j})_x + b(P_{i,j})_y + c(P_{i,j})_z)_{0 \le \substack{i \\ j} < 3}}{2} \right\rceil - \left\lceil \frac{max(|a|, |b|, |c|)}{2} \right\rceil \qquad (4\text{-}11)$$

4.5. Connectivity issues

4.5.1. Connectivity of digitized Bézier curves

The construction proposed previously for discretizing a 2D Bézier arc actually builds a continuous polygonal line (the dash-dotted line of Figure 4-2) whose discrete counterpart is the discretization of the Bézier curve. Thus the overall discretization forms an 8-connected path which is the most reasonable requirement for the discretization of a perfectly general Bézier arc. In simple cases, if the Bézier arc does not have too important changes in orientation, its overall discretization forms a 8-connected simple curve, i.e., every point of the discretization has exactly two 8-neighbors except for the endpoints (if the curve is not closed).

In the case of 3D Bézier curves the proposed construction does not create a continuous 3D polygonal 3D line since the midlines of consecutive segments may not be coplanar. However it can be shown that the overall discretization is still 26-connected.

Theorem 4-5. *Let D_1 and D_2 be two digital 3D lines as defined in Equation 3-29 and let H_1 and H_2 be their respective real square-based enclosing cylinders. If $H_1 \cap H_2 \neq \varnothing$ then D_1 and D_2 intersect or have two 26-adjacent points*

Proof. If $H_1 \cap H_2 \neq \varnothing$ then that intersection contains at least a real point $P_0(x_o, y_0, z_0)$. Since P_0 is contained in the enclosing cylinder H_1 of D_1, D_1 contains an integer point $I_1(X_1, Y_1, Z_1)$ such that

$$\begin{cases} |X_1 - x_0| < 1 \\ |Y_1 - y_0| < 1 \\ |Z_1 - z_0| < 1 \end{cases} \tag{4-12}$$

A similar statement holds for D_2 which contains an integer point $I_2(X_2, Y_2, Z_2)$ such that

$$\begin{cases} |X_2 - x_0| < 1 \\ |Y_2 - y_0| < 1 \\ |Z_2 - z_0| < 1 \end{cases} \tag{4-13}$$

Hence I_1 and I_2 are at least 26-adjacent if not equal which proves the theorem. ■

Each segment of 3D digital line making up the discretization of a 3D Bézier curve is the set of points contained in a truncated square-based cylinder (see Proposition 3-1 in Section 3.3).

Consecutive cylinders always have a non-void intersection since they contain at least one common real point (the point that is common to the two consecutive Bézier control polygons which is also part of the Bézier arc itself). Thus, thanks to Theorem 4-5, 26-adjacency between consecutive discrete straight line segments is guaranteed.

4.5.2. Connectivity of digitized Bézier surface patches

The connectivity problem becomes more intricate in the case of the digitization of Bézier surface patches where discrete connectivity has to be controlled along the whole length of the edges of patches in order to ensure that the resulting discretization is *well voxelized* in the sense of [11]. This involves determining the appropriate bounds of the equations defining the discrete plane patch in Equation 4-8. We examine the problem in the case of bicubic tensor product Bézier patches though the results still hold in the case of Bézier triangles. The connectivity problem of surfaces defined by adjacent discrete polygons has been studied by Andrès et al. [4] in the case of thick tunnel-free polygons, it still remains mostly open in the case of polygons based on naive digital planes and we only provide directions towards formal proofs.

Let B and B' be two adjacent Bézier bicubic tensor-product patches verifying Theorem 4-3 and having for respective control nets $(P_{i,j})_{\substack{0 \le i \le 3 \\ j}}$ and $(P'_{i,j})_{\substack{0 \le i \le 3 \\ j}}$ such that $P_{3,i} = P'_{0,i}$ for $0 \le i \le 3$ (Figure 4-3).

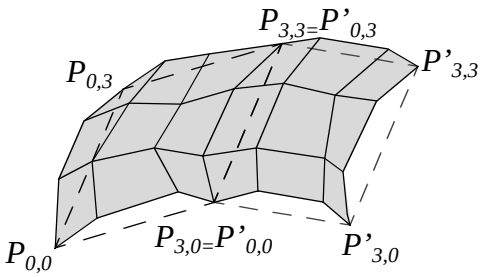


Figure 4-3: Adjacent bicubic Bézier control nets

Let $Q(a, b, c, \gamma, (\mathbf{q}_k, \lambda_k)_{0 \le k \le 3})$ and $Q'(a', b', c', \gamma', (\mathbf{q}'_k, \lambda'_k)_{0 \le k \le 3})$ be two naive digital plane patches and let us denote H and H' their real enclosing parallelepipeds defined in Equation 4-3. We consider two different cases depending upon whether (a, b, c) and (a', b', c') have the same main direction.

1. (a, b, c) **and** (a', b', c') **have the same main direction.** Let us assume that this direction is Oz. This condition writes $c = max(|a|, |b|, |c|)$ and $c' = max(|a'|, |b'|, |c'|)$. In that case Q and Q' are adjacent and their union is a 26-connected discrete surface iff $Q \cup Q$' still has the property of functionality of the individual digital plane patches, which amounts to say that

$H \cup H'$ has everywhere a thickness of 1 in the direction *Oz* and one full side of *H* and *H'* is contained in their intersection $H \cap H'$ (see Figure 4-4).

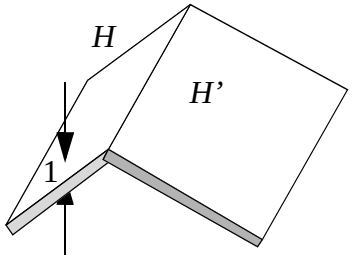


Figure 4-4: Union of the convex hulls of two discrete plane patches

A problem arises in the case of tensor-product surfaces which may not be planar even though they verify Theorem 4-3. This results in the choice of inconsistent normal values between adjacent patches that make it difficult to meet the previous connectivity criterion. Such a problem does not occur with triangles, since triangular patches are always planar. Hence, one solution is to divide the tensor-product patches (which are quadrilaterals) into two triangles and work only with discrete triangles.

2. (a, b, c) **and** (a', b', c') **have different main directions.** In the case where (a, b, c) and (a', b', c') have different main directions the previous condition does not hold anymore and we can only guarantee that $Q \cup Q'$ is 26-connected if $H \cap H'$ contains at least one of the sides of *H* or *H'* and $H \cup H'$ has everywhere a thickness of at least 1 with respect to the main directions of (a, b, c) or (a', b', c').

4.6. Summary

In this section we have presented a method to polygonalize Bézier curves and surfaces into discrete lines and plane patches. Unlike existing rendering algorithms that all require an arbitrary tolerance constant ε, our approach is entirely based on the geometry of the manipulated objects and theorems of discrete geometry. Moreover our termination criterion for the subdivision of Bézier curves is optimal. The criterion, though presented here in the case of cubic curves and bicubic surfaces, is general and can be extended to Bézier curves and surfaces of higher degree. Appropriate connectivity of Bézier curves polygonalized by our method is ensured both in 2D and 3D and we also provide a criterion to ensure appropriate separability of adjacent discrete surface patches in restricted cases. Further work is needed to determine a more general solution to this separability problem of 3D patches.

5 Multi-Scale Discrete Geometry

This chapter deals with the relations between discrete geometrical objects considered at different levels of resolution. It tries to answer questions like "what does the equation of a digital line become when this line is plunged into a lower resolution grid (subgroup of $\mathbb{Z}^2$)" ? What can be said of a digital parallelogram in the same circumstances ? How are these discrete objects covered by that grid ?

5.1. Introduction

Studying discrete objects like digital lines or planes at different levels of resolution is a recurring problem that has been little studied in the literature. And yet as the second half of this work will show (Section 7.3.4) there is a fundamental need for multi-scale geometry. Indeed applications that choose to use discrete geometrical objects instead of continuous models often need to work at a very high resolution to ensure acceptable precision. This in turn implies manipulating objects with thousands to billions of pixels. Given the capacities of today's computers, the only reasonable way to work with such large structures is to split them into parts. A common way of doing such a split is to divide the structure spatially into tiles, thereby introducing a new level of discreteness. Naturally, this new tiled structure needs to be related to the initial discrete structure.

Establishing a relation between a discrete geometric object at various discreteness scales is also fundamental from a theoretical point of view: it provides the ground to establish an equivalence between the digitizations at different scales of a continuous object and the continuous object itself.

5.2. Covering of a naive digital line by a lower resolution grid

A simple illustration of the multi-scale discreteness problem consists in finding the covering of a discrete line by a lower resolution grid. Let $D(a, b, \gamma)$ be a naive digital line such that $0 < a < b$ and $gcd(a, b) = 1$. Let us consider the subgroup $\mathcal{S}(h, v) = (\lambda h, \mu v)$ of $\mathbb{Z}^2$ (where λ, μ, h, v all are integers). Obviously the fundamental domain $[0, h) \times [0, v)$ of $\mathcal{S}(h, v)$ and its translations by the vectors $X\begin{pmatrix} h \\ 0 \end{pmatrix} + Y\begin{pmatrix} 0 \\ v \end{pmatrix}$ for $X, Y \in \mathbb{Z}$ induce a tiling of $\mathbb{Z}^2$ where each tile contains exactly one point of $\mathcal{S}(h, v)$ (for which reason we will refer indifferently to the tile itself or to the point of $\mathcal{S}(h, v)$ it contains). We are interested in the set of tiles of $\mathcal{S}(h, v)$ that intersect $D(a, b, \gamma)$ (Figure 5-1). This set of tiles is naturally a discrete line in the subgroup $\mathcal{S}(h, v)$ which can thus be described by means of an equation similar to Equation 2-4. We denote this line with Δ and call it the covering line of $D(a, b, \gamma)$ in $\mathcal{S}(h, v)$.

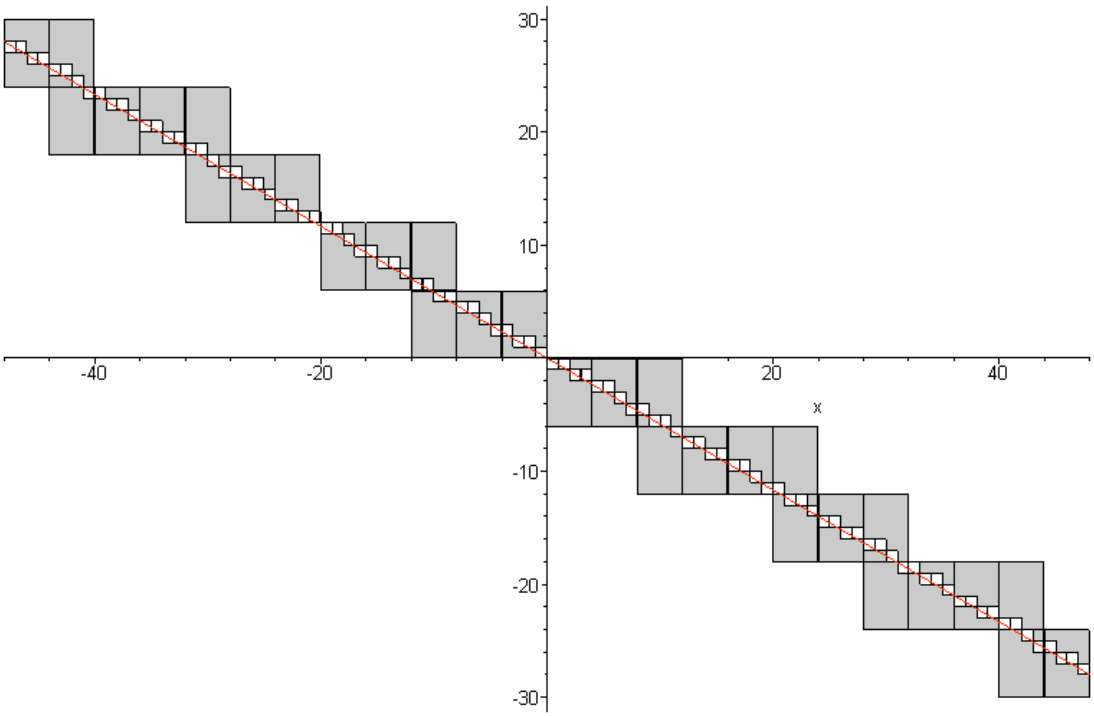


Figure 5-1: A discrete line covered by a lower resolution grid

The discrete line $D(7, 12, -12)$ is represented here with white squares together with the euclidean line $7x + 12y = 0$. The gray squares represent the covering of the discrete line by the tiling induced by the $\mathbb{Z}^2$ subgroup $(4\lambda, 6\mu)$ where $(\lambda, \mu) \in \mathbb{Z}$

The tiling generated by $\mathcal{S}(h, v)$ on $\mathbb{Z}^2$ induces a new coordinate system where coordinates (X, Y) are related to the canonical coordinates of $\mathbb{Z}^2$ by the following obvious relation:

$$\begin{aligned} X &= \left\lceil \frac{x}{h} \right\rceil \\ Y &= \left\lceil \frac{y}{v} \right\rceil \end{aligned} \tag{5-1}$$

which can be inverted as follows:

$$\begin{aligned} x &= hX + \left\{ \frac{x}{h} \right\} \\ y &= vY + \left\{ \frac{y}{v} \right\} \end{aligned} \tag{5-2}$$

By definition, $D(a, b, \gamma)$ can be written as

$$\gamma \leq ax + by < \gamma + b \tag{5-3}$$

Hence the equation of Δ in the coordinate system related to $\mathcal{S}$ writes:

$$\gamma - a\left\{\frac{x}{h}\right\} - b\left\{\frac{y}{v}\right\} \leq ahX + bvY < \gamma + b - a\left\{\frac{x}{h}\right\} - b\left\{\frac{y}{v}\right\} \tag{5-4}$$

Moreover, from Equation 5-3 stems

$$y = -\left\lfloor\frac{ax - \gamma}{b}\right\rfloor \tag{5-5}$$

and finally

$$\gamma - a\left\{\frac{x}{h}\right\} - b\left\{\frac{-\left\lfloor\frac{ax-\gamma}{b}\right\rfloor}{v}\right\} \leq ahX + bvY < \gamma + b - a\left\{\frac{x}{h}\right\} - b\left\{\frac{-\left\lfloor\frac{ax-\gamma}{b}\right\rfloor}{v}\right\} \tag{5-6}$$

In order to simplify Equation 5-6 let us introduce

$$\begin{aligned} m_x &= \left\{\frac{x}{h}\right\} \\ m_y &= \left\{\frac{y}{v}\right\} = \left\{\frac{-\left\lfloor\frac{ax-\gamma}{b}\right\rfloor}{v}\right\} \end{aligned} \tag{5-7}$$

Considering the fact that m_x and m_y vary when x steps through $\mathbb{Z}$, Equation 5-6 becomes:

$$\gamma - max_{x \in \mathbb{Z}}(am_x + bm_y) \leq ahX + bvY < \gamma + b - min_{x \in \mathbb{Z}}(am_x + bm_y) \tag{5-8}$$

Now to fully determine this equation that defines the covering of the digital line by the tiling, the exact range of $am_x + bm_y$, i.e., the values of $min_{x \in \mathbb{Z}}(am_x + bm_y)$ and $max_{x \in \mathbb{Z}}(am_x + bm_y)$, need to be calculated.

5.2.1. Determination of the range of $am_x + bm_y$

By definition in Equation 5-7, it is clear that:

$$\begin{aligned} 0 \leq m_x \leq h - 1 \\ 0 \leq m_y \leq v - 1 \end{aligned} \tag{5-9}$$

using these bounds in Equation 5-8 suggests for Δ an equation of the form:

$$\gamma - a(h-1) - b(v-1) \leq ahX + bvY < \gamma + b \tag{5-10}$$

But since $ahX + bvY$ can only assume values that are multiples of $g = \gcd(ah, bv)$, the bounds of Equation 5-8 can be refined. Denoting $pg = ah + bv$, this equation becomes

$$\left(1 + \left\lfloor \frac{a+b+\gamma}{g} \right\rfloor - p\right) g \le ahX + bvY \le \left\lfloor \frac{\gamma+b-1}{g} \right\rfloor g \qquad \text{if } \left\{ \frac{a+b+\gamma}{g} \right\} \neq 0 \qquad (5\text{-}11)$$

or

$$\left(\left\lfloor \frac{a+b+\gamma}{g} \right\rfloor - p\right) g \le ahX + bvY \le \left\lfloor \frac{\gamma+b-1}{g} \right\rfloor g \qquad \text{if } \left\{ \frac{a+b+\gamma}{g} \right\} = 0 \qquad (5\text{-}12)$$

However since m_x and m_y are linked through Equation 5-7, the precise bounds of $am_x + bm_y$ when x steps through $\mathbb{Z}$, need to be determined. In fact, we show in what follows, that even though $am_x + bm_y$ does not reach the absolute bounds 0 and $a(h-1) + b(v-1)$ in some cases, Equations 5-11 and 5-12 always hold.

Inverting Equation 5-7 yields

$$\begin{aligned} x &= kh + m_x \qquad k \in \mathbb{Z} \\ y &= lv + m_y \qquad l \in \mathbb{Z} \end{aligned} \qquad (5\text{-}13)$$

hence, thanks to Equation 5-5

$$-\left\lfloor \frac{a(kh + m_x) - \gamma}{b} \right\rfloor = lv + m_y \qquad (5\text{-}14)$$

which is equivalent to:

$$\gamma - kah - lbv \le am_x + bm_y < \gamma - kah - lbv + b \qquad (5\text{-}15)$$

$kah + lbv$ has for values the multiples of $g = \gcd(ah, bv)$:

$$kah + lbv = -tg \qquad t \in \mathbb{Z} \qquad (5\text{-}16)$$

Equation 5-15 can thus be rewritten as

$$\gamma + tg \le am_x + bm_y < \gamma + tg + b \qquad t \in \mathbb{Z} \qquad (5\text{-}17)$$

Equation 5-17 describes a family of digital lines of direction (a, b) parameterized by $t \in \mathbb{Z}$, which we denote $\mathcal{D}_t$. The intersection of $\mathcal{D}_t$ with the domain of (m_x, m_y), $[0, h) \times [0, v)$, defines the possible pairs (m_x, m_y) and therefore the range of possible values for the sum $am_x + bm_y$ (see Figure 5-2).

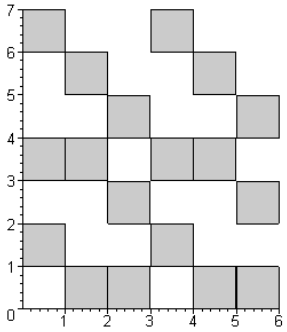


Figure 5-2: Determination of the range of $7m_x + 9m_y$

The gray squares represent the family of digital lines defined by $6 + 21t \le 7x + 9y < 6 + 21t + 9$ restricted to $[0, 6) \times [0, 7)$ and hence the possible values of (m_x, m_y)

Let us denote with (n_x, n_y) the pair of $\mathcal{D}_t \cap [0, h) \times [0, v)$ that yields the maximum value for the sum $an_x + bn_y$, i.e.

$$an_x + bn_y = max_{(m_x, m_y) \in \mathcal{D}_t \cap [0, h) \times [0, v)}(am_x + bm_y) \tag{5-18}$$

Let us also denote with δ^+ the difference between $am_x + bm_y$ evaluated at $(h-1, v-1)$ and at (n_x, n_y):

$$\delta^+ = a(h-1) + b(v-1) - (an_x + bn_y) \tag{5-19}$$

By construction it is clear that $0 \le \delta^+ < g$, therefore using Equation 5-17 and Equation 5-19 we can write that there exists one value of $t \in \mathbb{Z}$ such that:

$$0 \le a(h-1) + b(v-1) - \gamma - tg < g \tag{5-20}$$

Determining the maximum value of t verifying Equation 5-20 provides a more precise expression of the upper bound of δ^+. Equation 5-20 becomes

$$0 \le (p-t)g - (a + b + \gamma) < g \tag{5-21}$$

which can be further transformed:

$$0 \le \left(p - t - \left\lfloor \frac{a + b + \gamma}{g} \right\rfloor \right) g - \left\{ \frac{a + b + \gamma}{g} \right\} < g \tag{5-22}$$

then

$$p - t - \left\lceil \frac{a+b+\gamma}{g} \right\rceil + \left\lceil \frac{-\left\{ \frac{a+b+\gamma}{g} \right\}}{g} \right\rceil = 0 \tag{5-23}$$

At this point, we must distinguish two cases according to $\left\{ \frac{a+b+\gamma}{g} \right\}$.

1. g **divides** $a+b+\gamma$**.** In this case Equation 5-23 becomes

$$t = p - \left\lceil \frac{a+b+\gamma}{g} \right\rceil \tag{5-24}$$

Hence δ^+ is bounded by

$$0 \le \delta^+ \le pg - \gamma - a - b - \left(p - \left\lceil \frac{a+b+\gamma}{g} \right\rceil \right) g \tag{5-25}$$

And finally we get $\delta^+ = 0$ which means that $(n_x, n_y) = (h-1, v-1)$ and that the lower bound of Equation 5-12 holds.

2. g **does not divide** $a+b+\gamma$**.** In this case Equation 5-23 becomes

$$t = p - \left\lceil \frac{a+b+\gamma}{g} \right\rceil - 1 \tag{5-26}$$

Hence δ^+ is bounded by

$$0 \le \delta^+ \le g - \left\{ \frac{a+b+\gamma}{g} \right\} \tag{5-27}$$

It follows:

$$\gamma - a(h-1) + b(v-1) \le \gamma - (an_x + bn_y) \le \gamma - a(h-1) + b(v-1) + g - \left\{ \frac{a+b+\gamma}{g} \right\} \tag{5-28}$$

which is equivalent to:

$$-pg + \gamma + a + b \le \gamma - (an_x + bn_y) \le \left(1 - p + \left\lceil \frac{a+b+\gamma}{g} \right\rceil \right) g \tag{5-29}$$

showing that the lower bound of Equation 5-11 holds.

A similar reasoning can be applied with the minimum value of $am_x + bm_y$ throughout $\mathcal{D}_t \cap [0, h) \times [0, v)$ which shows that the upper bounds of Equations 5-11 and 5-12 also hold in all cases. Those equations can be formulated as a single expression which yields the following theorem:

Theorem 5-1. *The discrete line* Δ *of* $\mathcal{S}(h, v)$ *covering the naive digital line* $D(a, b, \gamma)$ *of* $\mathbb{Z}^2$ *is defined by:*

$$-\left\lceil \frac{-(\gamma + a + b)}{g} \right\rceil - \alpha - \beta \leq \alpha X + \beta Y < \left\lceil \frac{\gamma + b - 1}{g} \right\rceil + 1 \qquad (5\text{-}30)$$

where $g = \gcd(ah, bv)$, $\alpha = ah/g$ *and* $\beta = bv/g$

As an example, let us study the covering of the digital line

$$D(7, 9, 6): \qquad 6 \leq 7x + 9y < 15 \qquad (5\text{-}31)$$

by the tiling induced by $\mathcal{S}(6, 7)$. The equation of the covering line Δ is given by Equation 5-30:

$$\Delta: \qquad -3 \leq 2X + 3Y < 1 \qquad (5\text{-}32)$$

which we can see in Figure 5-3.

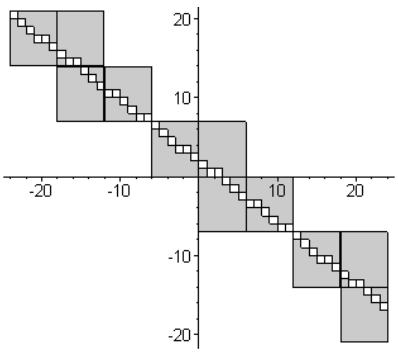


Figure 5-3: The covering line of $D(7, 9, 6)$ **by** $\mathcal{S}(6, 7): -3 \leq 2X + 3Y < 1$

5.3. Covering of a parallelogram by a low resolution grid

5.3.1. Introduction

We will now present a solution to another problem related to multiple scales of discreteness: given a digital parallelogram $(ABCD)$ in $\mathbb{Z}^2$, how does one find its covering by a tiling $\mathcal{S}(h, v) = (\lambda h, \mu v)$ where $\lambda, \mu, h, v \in \mathbb{Z}$ (Figure 5-4) ? This question actually originated from

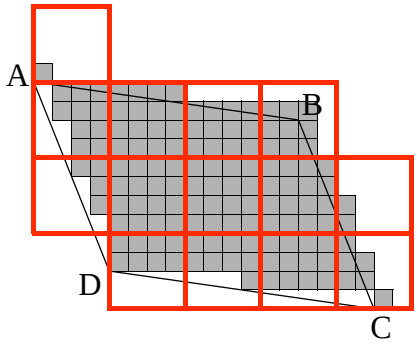


Figure 5-4: Covering of a digital parallelogram by a tiling

Gray squares represent the pixels making up the digital parallelogram (remember our convention roots pixels at their bottom-left corner, see Section 2.1). Transparent thick squares represent the tiles that cover the digital parallelogram.

a very practical concern and the solution presented here finds its way into the digital plane and surface extraction algorithms introduced in Chapters 7 and 8.

More precisely, let $A\begin{pmatrix}x_A\\y_A\end{pmatrix}$, $B\begin{pmatrix}x_B\\y_B\end{pmatrix}$, $C\begin{pmatrix}x_C\\y_C\end{pmatrix}$ and $D\begin{pmatrix}x_D\\y_D\end{pmatrix}$ be four integer points, we define the integer parallelogram $\Pi(A, B, C, D)$ as the set of integer points contained within the euclidean parallelogram defined by the four points (including the boundary lines) denoted $\mathcal{P}(A, B, C, D)$. This corresponds to the intersection of two thick digital lines which can be formulated as follows:

$$\begin{aligned} n_A \le n_x x + n_y y < n_D + 1 \\ p_A \le p_x x + p_y y < p_B + 1 \end{aligned} \qquad (5\text{-}33)$$

where $\mathbf{n}\begin{pmatrix}n_x\\n_y\end{pmatrix}$ is the vector normal to **AB** pointing to the interior of the parallelogram, i.e. having the same orientation as **AD**, $\mathbf{n} \cdot \mathbf{AD} > 0$, and $\mathbf{p}\begin{pmatrix}p_x\\p_y\end{pmatrix}$ is the vector normal to **AD** also pointing to the interior of the parallelogram, i.e. having the same orientation as **AB**, $\mathbf{p} \cdot \mathbf{AB} > 0$. Moreover, $n_A = n_x x_A + n_y y_A$, $n_D = n_x x_D + n_y y_D$, $p_A = p_x x_A + p_y y_A$ and $p_B = p_x x_B + p_y y_B$. We also suppose the parallelogram is not degenerate, i.e. $n_D - n_A + 1 \ge max(|n_x|, |n_y|)$ and $p_B - p_A + 1 \ge max(|p_x|, |p_y|)$. We are interested in determining the intersection of $\Pi(ABCD)$ with the lattice $\mathcal{S}(h, v)$.

Since a digital parallelogram is defined as nothing more than the intersection of two digital lines, the previous method used to determine the covering of a line could also be applied and extended for this particular situation. However the demonstration is a bit tricky, so we will use

a different approach here which also illustrates the wealth of possibilities available to solve problems in discrete geometry.

5.3.2. Morphological dilation

This approach is based on the concept of dilation by a structuring element found in mathematical morphology, itself issued from Minkowski's addition [14][50]. This operation is defined as follows.

Definition 5-1. Let E be a subset of $\mathbb{R}^2$. At each point x of $\mathbb{R}^2$, called *guiding point*, we define a subset B_x of $\mathbb{R}^2$ called *structuring element*. We call *dilation of E by the structuring element B_x*, the subset of $\mathbb{R}^2$:

$$E \oplus B_x = \{x \in \mathbb{R}^2 / B_x \cap E \neq \varnothing\} \tag{5-34}$$

That is, for each point x of $\mathbb{R}^2$, the question "does B_x touch the set E ?" is asked. The set of points E' for which the answer to this question is positive is the dilation of E (Figure 5-5).

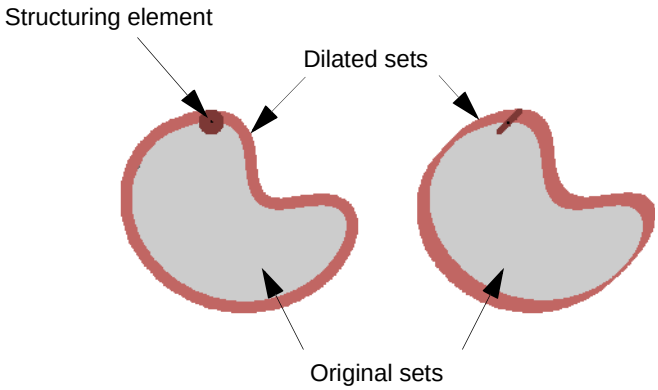


Figure 5-5: Dilation by two different centered structuring elements

There is no problem in applying a similar definition in $\mathbb{Z}^2$. Let us then consider as a structuring element a rectangle rooted at its bottom left corner: $R_{(x, y)} = [x, x + h) \times [y, y + v)$. $R_{(0, 0)}$ corresponds to the fundamental domain of the lattice $\mathcal{S}(h, v)$. Let $\Pi'(A, B, C, D)$ be the result of the dilation of $\Pi(A, B, C, D)$ by $R_{(x, y)}$ in $\mathbb{Z}^2$:

$$\Pi'(A, B, C, D) = \Pi(A, B, C, D) \oplus R_{(x, y)} \tag{5-35}$$

It is clear that the tile of $\mathcal{S}(h, v)$ associated to the point (x, y) is identical to the structuring element $R_{(x, y)}$. Therefore, right from the definition of the dilation operation, it becomes obvious that the tiles of $\mathcal{S}(h, v)$ that make up the covering of $\Pi(A, B, C, D)$ are the tiles associated with the points of the lattice $(\lambda h, \mu v)$, $(\lambda, \mu \in \mathbb{Z})$ contained in $\Pi'(A, B, C, D)$. Indeed any tile

of $\mathcal{S}(h, v)$ associated with a point outside of $\Pi'(A, B, C, D)$ does not intersect $\Pi(A, B, C, D)$ while every tile associated with a point inside of $\Pi'(A, B, C, D)$ intersects $\Pi(A, B, C, D)$ (Figure 5-6).

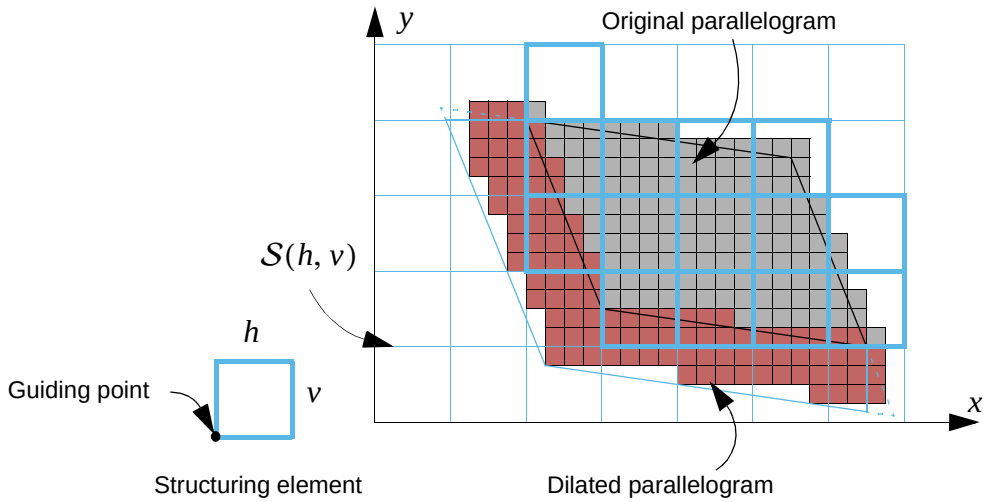


Figure 5-6: Dilation and covering of a digital parallelogram

The original digital parallelogram consists of the gray pixels. The dilated parallelogram consists of the gray pixels and the added brown pixels. The euclidean boundaries of both digital parallelograms are shown as continuous lines. The covering of the original parallelogram by the blue tiling is represented by thick transparent blue squares.

Figure 5-6 suggests that a formal algebraic expression of the dilation of $\Pi(A, B, C, D)$ can be derived as a truncated digital parallelogram (the dashed lines in Figure 5-6 show where this truncation takes place). To establish this expression, a preliminary result is needed however.

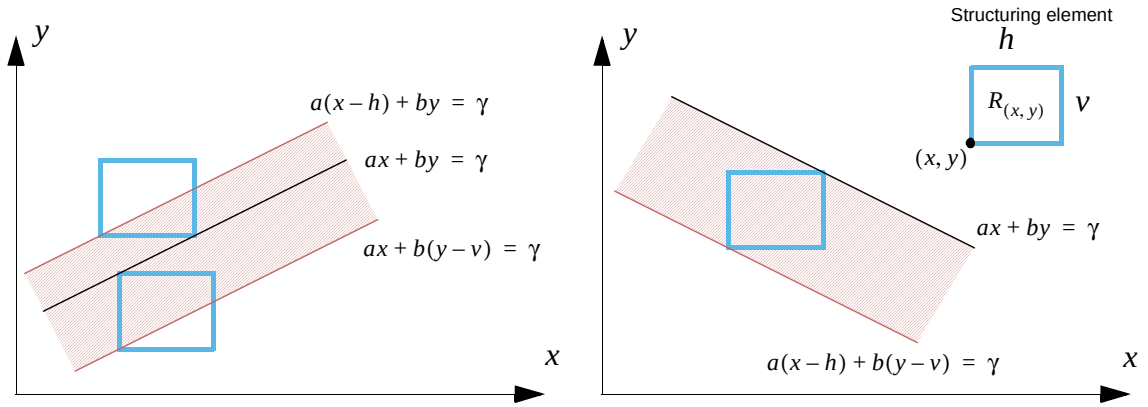
5.3.3. Dilation of an euclidean line by a rectangle

Let D be an euclidean line of rational direction defined by the following equation:

$$D: \quad ax + by = \gamma \tag{5-36}$$

We are interested in determining the equations of its dilation by a rectangle $R_{(x, y)} = [x, x + h] \times [y, y + v]$. This dilation is naturally a strip, i.e., a region of $\mathbb{R}^2$ contained between two euclidean lines parallel to D.

As illustrated by Figure 5-7, two different cases must be distinguished.



(a) a and b have different signs (b) a and b have the same sign

Figure 5-7: Dilation of an euclidean line by a rectangle

1. a, b **have different signs.** In this case the boundary lines of the dilated region are given by the two equations:

$$ax + by = \gamma - ah \tag{5-37}$$

$$ax + by = \gamma - bv \tag{5-38}$$

To order these two bounds a further distinction must be made.

- If $a < 0$ and $b > 0$, the equation of the strip writes:

$$\gamma - bv \le ax + by \le \gamma - ah \tag{5-39}$$

- If $a > 0$ and $b < 0$, the equation of the strip writes:

$$\gamma - ah \le ax + by \le \gamma - bv \tag{5-40}$$

2. a, b **have the same sign (or one of the two is null).** In this case one the boundary lines of the dilated region is the line itself while the other one is given by the equation:

$$ax + by = \gamma - ah - bv \tag{5-41}$$

Again, to order the two bounds into a single equation, two cases must be distinguished:

- If $a < 0$ or $b < 0$, the equation of the strip writes:

$$\gamma \le ax + by \le \gamma - ah - bv \tag{5-42}$$

- If $a > 0$ or $b > 0$, the equation of the strip writes:

$$\gamma - ah - bv \le ax + by \le \gamma \tag{5-43}$$

Equations 5-39, 5-40, 5-42 and 5-43 can then be summarized as follows:

$$\begin{aligned}
\gamma - bv \le ax + by \le \gamma - ah \qquad & a < 0 \wedge b > 0 \\
\gamma - ah \le ax + by \le \gamma - bv \qquad & a > 0 \wedge b < 0 \\
\gamma \le ax + by \le \gamma - ah - bv \qquad & (a \le 0 \wedge b > 0) \vee (a < 0 \wedge b \le 0) \\
\gamma - ah - bv \le ax + by \le \gamma \qquad & (a \ge 0 \wedge b > 0) \vee (a > 0 \wedge b \ge 0)
\end{aligned} \tag{5-44}$$

By further noticing that

$$\frac{\alpha + |\alpha|}{2} = \begin{cases} \alpha & \text{if } (\alpha \ge 0) \\ 0 & \text{if } (\alpha \le 0) \end{cases} \tag{5-45}$$

and that

$$\frac{\alpha - |\alpha|}{2} = \begin{cases} \alpha & \text{if } (\alpha \le 0) \\ 0 & \text{if } (\alpha \ge 0) \end{cases} \tag{5-46}$$

the Equations 5-44 can be combined together into the following single expression:

Theorem 5-2. *The dilation in $\mathbb{Z}^2$ of the euclidean line D: $ax + by = \gamma$ by a rectangle $R_{(x,y)} = [x, x+h] \times [y, y+v]$ is the set of integer points verifying the following diophantine inequation:*

$$\gamma - \frac{ah + bv + |ah| + |bv|}{2} \le ax + by \le \gamma - \frac{ah + bv - |ah| - |bv|}{2} \tag{5-47}$$

5.3.4. Morphological dilation of a digital parallelogram

Thus far we have only considered the dilation of euclidean objects of $\mathbb{R}^2$. Now we want to consider the dilation of a digital parallelogram of $\mathbb{Z}^2$ by a rectangular structuring element $R_{(x,y)}$ as defined in section 5.3.2. Transposing the results established previously to this latter case involves some subtleties.

By definition, the dilation of the digital parallelogram $\Pi(A, B, C, D)$ by $R_{(x,y)}$ in $\mathbb{Z}^2$, denoted $\Pi'(A, B, C, D)$, is the set of integer points (x, y) such that $\Pi(A, B, C, D)$ and $R_{(x,y)}$ share at least one common integer point. $\Pi(A, B, C, D)$ is defined by the system of Equation 5-33 which means that it is the set of integer points contained within the region delimited by four euclidean boundary lines, i.e., the euclidean parallelogram $\mathcal{P}(A, B, C, D)$.

For $\Pi(A, B, C, D)$ and $R_{(x,y)}$ to share at least one integer point, $\mathcal{P}(A, B, C, D)$ and $R_{(x,y)}$ must have an overlap of width or height of one at least, in order to ensure that this overlapping zone contains at least one integer point. Hence $\Pi'(A, B, C, D)$ is equivalent to the set of integer points contained in the region delimited by the dilation, in $\mathbb{R}^2$, of the euclidean boundaries of

$\Pi(A, B, C, D)$ by a structuring element one unit smaller in x and y than $R_{(x, y)}$ which we denote $R'_{(x, y)} = [x, x + h - 1) \times [y, y + v - 1)$ (see Figure 5-8).

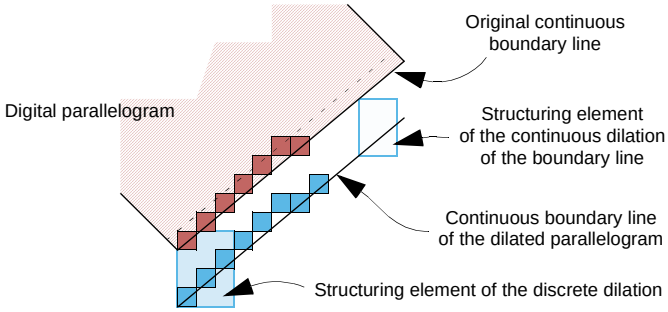


Figure 5-8: Discrete vs. continuous dilation

Thus we can define the discrete dilation of $\Pi(A, B, C, D)$ by $R_{(x, y)}$ by considering the continuous dilation of $\mathcal{P}(A, B, C, D)$ by $R'_{(x, y)}$ and applying Theorem 5-2 to each boundary line. This leads to the following equation system:

$$\begin{cases} n_A + \delta_n^- \le n_x x + n_y y < n_D + \delta_n^+ \\ p_A + \delta_p^- \le p_x x + p_y y < p_B + \delta_p^+ \end{cases} \tag{5-48}$$

where

$$\delta_i^- = -\frac{i_x(h-1) + i_y(v-1) + |i_x(h-1)| + |i_y(v-1)|}{2}$$
$$\delta_i^+ = -\frac{i_x(h-1) + i_y(v-1) - |i_x(h-1)| - |i_y(v-1)|}{2} + 1 \qquad i = n, p \tag{5-49}$$

The set defined by Equation 5-48 is actually larger than $\Pi'(A, B, C, D)$ (see Figure 5-6). Indeed, given the considered structuring element and by definition of the dilation operation, $\Pi'(A, B, C, D)$ must be entirely contained in a rectangular area defined by $(x_{min} - h, x_{max}] \times (y_{min} - v, y_{max}]$ where

$$\begin{aligned} x_{min} &= min(x_A, x_B, x_C, x_D) \\ x_{max} &= max(x_A, x_B, x_C, x_D) \\ y_{min} &= min(y_A, y_B, y_C, y_D) \\ y_{max} &= max(y_A, y_B, y_C, y_D) \end{aligned} \tag{5-50}$$

Therefore Equation 5-48 needs to be refined with the following additional conditions in order to yield the equation system defining precisely $\Pi'(A, B, C, D) = \Pi(A, B, C, D) \oplus R_{(x, y)}$:

$$\Pi'(A, B, C, D): \quad \begin{cases} n_A + \delta_n^- \le n_x x + n_y y < n_D + \delta_n^+ \\ p_A + \delta_p^- \le p_x x + p_y y < p_B + \delta_p^+ \\ x_{min} - h < x \le x_{max} \\ x_{min} - v < y \le x_{max} \end{cases} \tag{5-51}$$

By further scaling these equations by (h, v), we obtain a system that can be solved by the method exposed in Section 2.5.1.

Theorem 5-3. *The covering of the parallelogram* $\Pi(A, B, C, D)$ *by the tiling* $\mathcal{S}(h, v)$ *is the set of tiles of coordinates* (x, y) *(in the coordinates system associated to* $\mathcal{S}(h, v)$*) verifying:*

$$\begin{cases} n_A + \delta_n^- \le n_x hx + n_y vy < n_D + \delta_n^+ \\ p_A + \delta_p^- \le p_x hx + p_y vy < p_B + \delta_p^+ \\ x_{min} - h < hx \le x_{max} \\ y_{min} - v < vy \le y_{max} \end{cases} \tag{5-52}$$

where $\delta_n^-, \delta_n^+, \delta_p^-, \delta_p^+, x_{min}, x_{max}, y_{min}, y_{max}$ *are defined in Equations 5-49 and 5-50.*

As an example, let us consider the parallelogram defined by the following 4 points, $A\begin{pmatrix}-12\\10\end{pmatrix}$, $B\begin{pmatrix}1\\18\end{pmatrix}$, $C\begin{pmatrix}20\\8\end{pmatrix}$, $D\begin{pmatrix}7\\0\end{pmatrix}$ equivalently defined by the following equation system:

$$\begin{cases} -226 \le 8x - 13y < 79 \\ -164 \le 10x + 19y < 1 \end{cases} \tag{5-53}$$

The covering of this parallelogram in $\mathcal{S}(6, 7)$ is given by Theorem 5-3 which writes in this case:

$$\begin{cases} -266 \le 48x - 91y < 135 \\ -94 \le 60x + 133y < 353 \\ -16 < 6x \le 20 \\ -31 < 7y \le 30 \end{cases} \tag{5-54}$$

The first two equations can be rewritten according to the method of Section 2.5.1:

$$\begin{pmatrix} -266 \\ -94 \end{pmatrix} \leq \begin{pmatrix} 1 & 0 \\ -4687 & 11844 \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} < \begin{pmatrix} 135 \\ 353 \end{pmatrix} \tag{5-55}$$

Solving this equation and applying the identity $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -36 & 91 \\ -19 & 48 \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix}$, yields the solution:

X	-235	-230	-187	-182	-139	-134	-91	-48	-43	0	5	48	53	96
Y	-93	-91	-74	-72	-55	-53	-36	-19	-17	0	2	19	21	38
x	-3	-1	-2	0	-1	1	0	-1	1	0	2	1	3	2
y	1	2	1	2	1	2	1	0	1	0	1	0	1	0

where one can notice that the $(-3, 1)$ solution (grayed out in the table) is not compatible with the two other equations in Equation 5-54:

$$\begin{cases} -2 \leq x \leq 3 \\ -4 \leq y \leq 4 \end{cases} \tag{5-56}$$

so it is discarded. The final result is illustrated in Figure 5-9.

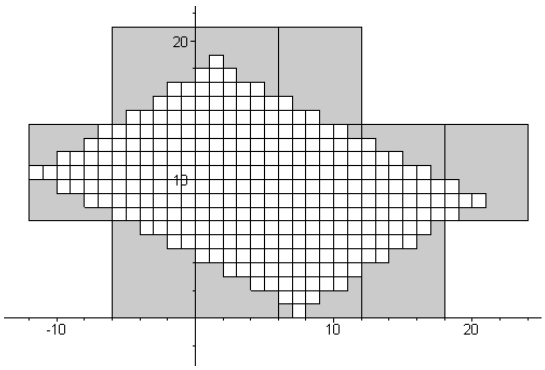


Figure 5-9: Covering of a digital parallelogram by a lower resolution grid

The parallelogram defined in Equation 5-53 is shown here with its covering by the tiling $\mathcal{S}(6, 7)$

5.4. Summary

In this section we have illustrated by means of two different examples some problems presented by multi-scale discrete geometry, i.e., what happens when discrete objects are considered with respect to rectangular subgroups of $\mathbb{Z}^2$ instead of $\mathbb{Z}^2$ itself. Such subgroups define tessel-

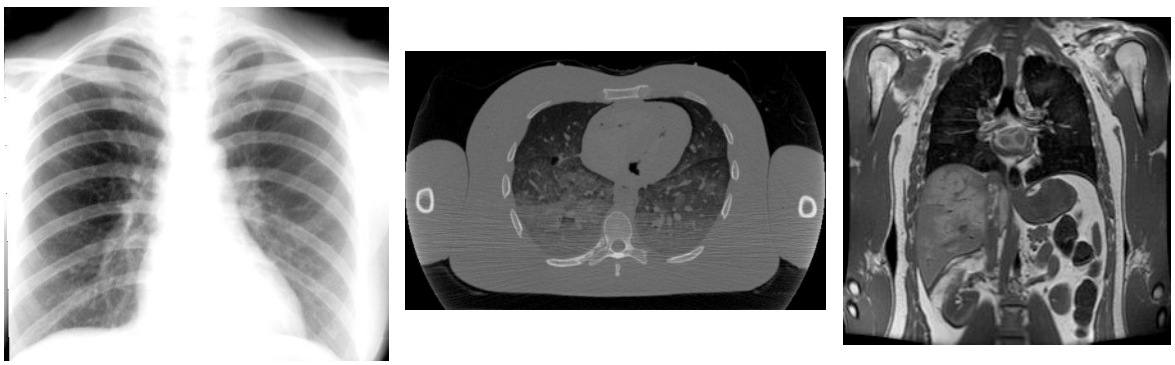
lations of the plane by rectangular tiles. Our primary concern was to determine the coverings of discrete objects by such tilings. Using two different approaches, one direct, based on the algebraic equation of a naive straight line and the second one geometric, based on the concept of morphological dilation, we established the equations defining these coverings exactly for digital lines and digital parallelograms. This latter result finds a direct concrete application in the digital plane extraction algorithm presented in Chapter 8.

Medical Image Visualization Applications

6 Introduction

6.1. Medical imaging techniques and challenges

Medical imaging techniques play an ever increasing role in patient care. Since the first medical use of X-rays at the beginning of this century, physicians have investigated a wide variety of physical processes to image the human body, some of them are of routine use for diagnosis today: magnetic resonance imaging (MRI), positron emission tomography (PET), single-photon emission computed tomography (SPECT) ultrasound imaging (echography) and naturally X-rays which remain the most commonly used modality.



X-ray radiography
Coronal view of the thorax

X-ray CT
Axial view of the thorax

MRI
Coronal view of the thorax

Figure 6-1: Sample medical images

But the real revolution was the introduction of *Computed Tomography* (CT) in the 70's [10]. Initially based on X-rays, CT is nowadays a technique involved in most of other imaging modalities (even though the term CT itself is often used as a shorthand for X-ray CT). Indeed a conventional radiography offers a projection of the absorption of a flux of X-rays by a region of the body, i.e., the image is the result of the cumulative absorption of rays by all the tissues traversed by the X-ray beam. On such an image the absorption of a single organ or tissue can not be estimated. CT on the other hand, brings the possibility to get a view of the individual absorption at every point of a thin slice of the examined region. This view is actually obtained by computation (hence the name computed tomography) and not by direct imaging. CT uses not one, but a series of radiographies taken at regular intervals around the region to examine (Figure 6-2). Thanks to the inverse Radon transform (backprojection), those individual projections are combined together to calculate the individual absorption at every point of the examined

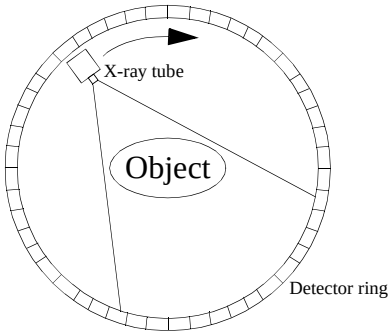


Figure 6-2: Principle of an X-ray CT scanner

slice thus providing a detailed image whose contrast depends on the variations of X-ray absorption by the body cells at every point of the slice.

Thus CT has opened the doors of hospitals to computers and nowadays all but a few medical images are in fact computer calculated images that are printed on film afterwards. Facilities offered by computer storage and archiving also contribute to this trend, and even the traditional X-ray radiographies will probably be replaced by a digital equivalent in the near future. The images produced today by the vast majority of medical imaging modalities (X-ray CT, MRI, SPECT...) are series of bi-dimensional digital images which are fed into computers for further exploitation. Stacking these 2D images forms a three-dimensional image of the examined region. Computer imaging techniques are then used to present physicians with selected views of the data. There exist essentially two main different ways to display 3D medical data [9, 18, 52]. *Surface rendering* was developed first and was very popular in the 70's and 80's. The current trend, however, seems to be more in favor of *volume rendering*.

Surface rendering aims at reducing the amount of data to process by deriving a lower dimensional representation of an isolated surface of an object of interest. This intermediate representation is generally a mesh of triangles but can also be made of a set of voxels or surfaces of voxels belonging to the surface of interest. Three-dimensional views of the surface can then be rendered very fast, sometimes using hardware acceleration. One of the major drawbacks of this approach is that the whole original volume must be reprocessed whenever a new organ or region of interest is selected.

On the other hand, volume rendering uses the original 3D volume directly thereby avoiding the volume reprocessing penalty [40]. However, volume rendering requires huge computing resources since it manipulates much more data than surface rendering algorithms. As computer power increases and the implementations of the algorithm become more efficient, the simpler concept of volume rendering and its higher quality images become more and more appealing.

6.2. High-performance visualization of planes and surfaces

As the resolution of acquisition devices increases, the amount of data acquired in a single examination becomes huge. Processing such volumes of information traditionally requires expensive parallel computer hardware. In the past, medical imaging has required mainframe computers or high-end workstations with multiple CPUs in order to achieve reasonable computation times. Several special purpose multiprocessor architectures and hardware based algorithm implementations have even been developed [52]. For these reasons, physicians used to content themselves with the sole 2D views and have become experts in the process of mentally reconstructing the actual volume from these images. Recent interviews with doctors have shown that most of them still feel more comfortable with two-dimensional views and have difficulties in manipulating and extracting relevant information from volumic images. 3D visualization, though spectacular, remains of limited medical interest. Thus, slicing, i.e. extracting a plane having any desired position and orientation from the acquired 3D volume remains the tool of choice in routine diagnosis and treatment.

These considerations and the contacts established with Professor O. Ratibe, Dr. L. Bidaut and Dr. R. Welz at the University Hospital of Geneva guided the design of the medical visualization applications that are presented in the following sections. Within this thesis, the *DigiPlan* library was developed for the extraction of slices of arbitrary orientation out of 3D biomedical acquisitions (CT, MRI). However, this type of view rapidly appears somewhat limited since rather few of the structures in the human body are planar. Often one would like to get a view like a slice taken throughout the whole length of the vertebral column or the jaw for instance. This led to the development of *DigiSurf*, a library that generalizes DigiPlan and generates a flat 2D view of ruled surfaces extracted from 3D medical image volumes.

The high resolution of medical scans is also synonym of high computer storage space requirements. Until recently, the only viable means of exchange of images between physicians was film. With the expansion of CD-ROM usage however, this situation is changing. Patients carrying their own examination images on CD-ROM from the clinic to their MD is an option being considered. Besides, thanks to the Internet, doctors can now download the images of their patients from the radiology center down to their own office for local examination or even browse them on-line. Telemedicine is a hot research topic and will certainly become a reality in the forthcoming years: physicians will be able to have virtual meetings with colleagues, exchange images of patients, have their diagnostic confirmed by distant specialists, etc... All this implies that modern medical visualization software must be highly scalable in order to be able to run on the low-end PC that can be found in medical practices as well as on the powerful Internet imaging servers located in hospitals and clinics.

Fortunately the increase in performance of commodity hardware such as Pentium-based PCs as well as the possibility of connecting arrays of disks through SCSI channels enables building much cheaper interactive 3D image storage and visualization devices. Striping the image onto a set of disks and reading from them in parallel offers an aggregate bandwidth compatible with the requirements of medical imaging. Furthermore, several PCs can be connected through com-

modity network hardware such as Ethernet or Fast-Ethernet and provide increased computing power for algorithms designed to run in parallel.

On the software side, visualization algorithms designed for such systems need to be able to run in parallel on multiple CPUs and achieve high efficiency in order not to choke when they are fed tens of megabytes of data per second read in parallel from the disks.

Unlike RISC processors that achieve great performance dealing with floating-point numbers, CPUs of the x86 family which form the core of PCs today, are processors largely intended for integer computation. MMX technology has even reinforced this trend. MMX (MultiMedia eXtensions), a technology developed by Intel, adds new instructions to the traditional x86 instruction set, especially targeted at signal processing. These instructions, inspired from the DSPs (Digital Signal Processors), use the floating-point registers of the x86 CPU in order to save chip space and to be able to process several integer operands in parallel. Switching contexts between MMX and floating-point register usage imposes a great penalty in terms of CPU cycles. Therefore, such an architecture clearly favors the choice of pure integer arithmetic.

Moreover 3D volumetric datasets produced by medical imaging modalities are inherently discrete, made of voxels. Mapping continuous models on such discrete spaces, though apparently simple, actually introduces several difficulties and inconsistencies. Discrete geometry, as it has been shown in the first half of this work, takes advantage of the discrete nature of such objects, considering them for what they are, working directly on them and not on a continuous pseudo-equivalent. This approach has incomparable advantages like consistency and efficiency which allow to derive numerically stable algorithms well suited for parallelization. These aspects make discrete geometry appear like the tool of choice in the design of biomedical visualization algorithms. Through the following application examples we will thus show that discrete geometry, far from being a pure abstraction, is also strongly rooted in concrete problems and that many theoretical results developed previously find direct applications.

6.3. The CAP/PS^2 parallelization framework

To achieve the best performance on a multi-PC multi-disk architecture, an algorithm needs to avoid data transfer overheads and must hide disk latency by pipelining the asynchronous disk accesses and the actual computation. Indeed, it is remarkable that even simple PCs, thanks to technologies like SCSI and DMA, can perform disk read/write operations in parallel with computation since the main processor is not used for data transfers between storage peripherals and memory. Developing parallel I/O intensive applications taking advantage of this fact and involving multiple processes running on different processors represents a very tedious effort. One needs to define and implement application-specific protocols in order to exchange parameters and data between different processors that do not share a common memory space. Furthermore, debugging parallel applications and ensuring they are deadlock-free remains a very difficult task. Ensuring that the program is portable from one parallel architecture to another and that it scales well introduces yet another constraint.

In order to facilitate the development of parallel applications, the Peripheral Systems Laboratory at EPFL (Dr. B. Gennart, and Dr. V. Messerli under the direction of Prof. R.D. Hersch) has developed the CAP/PS^2 (*Parallel Storage and Processing System*) parallelization framework [43]. This system targets architectures such as described above, consisting of a number of *storage and processing nodes* (per node: a processor connected to several disks) and a *client node* (a processor with an attached display). The client node and the storage/processing nodes are interconnected by a Fast Ethernet LAN (100 Mb/s) (Figure 6-3).

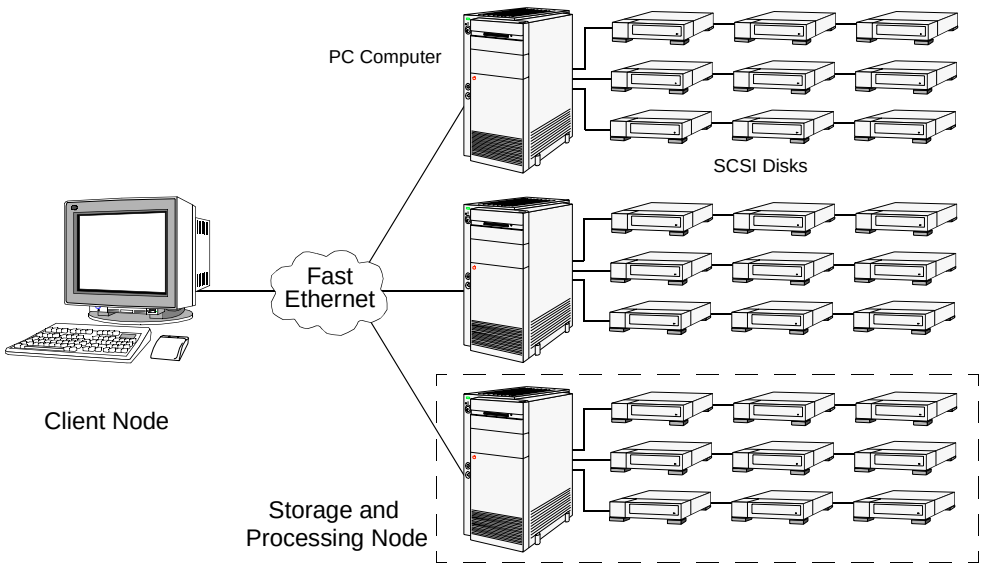


Figure 6-3: Hardware architecture targeted by the PS^2 parallelization framework

The core of the CAP/PS^2 framework comprises:

- a high-performance parallel file system built on top of the native MS Windows NT file system (NTFS) running on each node. This parallel file system declusters large files into sub-files that reside in the different storage and processing nodes and provides all the necessary primitives to access these subfiles;
- the CAP (Computer-Aided Parallelization) extension to C++ which enables application programmers to specify at a high-level of abstraction the flow of data between pipelined-parallel operations ensuring that the resulting program is deadlock-free, portable and appropriately combines parallel storage access routines and image processing operations [27, 28].

The *DigiPlan* and *DigiSurf* libraries were developed using this framework and thus take full advantage of the possibilities it offers :

1) Striping among several disks enables handling images of arbitrary size with optimum performance as data is fetched in parallel from the disks
2) Applications are scalable since they are able to run on platforms ranging from the simple mono-processor to a multiprocessor shared-memory system and up to a cluster of networked computers

7 DigiPlan: Volume Slicing with Discrete Planes

This chapter presents an algorithm for extracting planar slices of arbitrary orientation from a 3D voxel-based volume. Discrete geometry is shown to be well-suited for this application. It simplifies the parallelization of the algorithm and contributes to its overall efficiency. A brief summary of the performance of the algorithm running on various parallel configurations is also presented. Finally, a concrete Web application using this algorithm for medical imaging is also described.

7.1. Introduction

For years, physicians and radiologists have been used to reading directly the printouts of the slice images produced by various medical imaging modalities (CT, IRM, etc...). This kind of slice series is largely used for the education of medicine students and thus young doctors get familiar very early with the interpretation of this type of two-dimensional views. And though three-dimensional imaging is now entering medical schools, most physicians still feel more comfortable with the examination of two-dimensional slice views. Traditionally, medical imaging devices provide three types of orthographic views (*axial*, *sagittal* and *coronal*) which still make up the basis for most diagnoses.

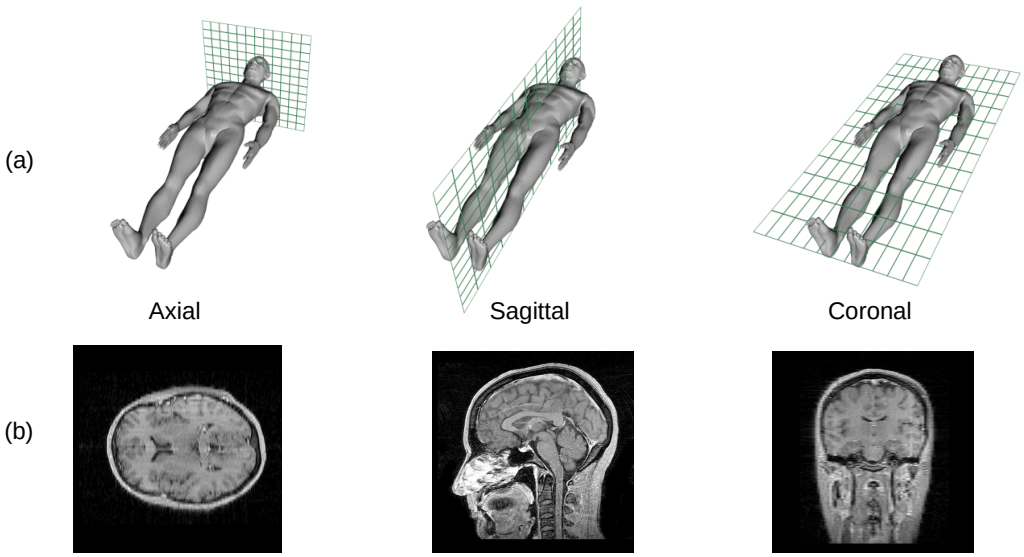


Figure 7-1: Traditional orientations of biomedical slice images

(a) Human 3D model, courtesy of Julien Proux; (b) MRI head scan, courtesy of University Hospital of Lausanne (CHUV)

This limited set of orientations is convenient since it always offers the same viewpoint, making it simpler for the physicians to recognize precisely which anatomical structure they are watching and also giving them a solid basis for comparison between similar shots, say, an healthy organ and a pathology of this organ or shots of the same anatomical region taken at different times.

Orthographic views are sometimes too restrictive and breaking this limitation on orientation to offer arbitrary oblique views is a natural extension. Rhodes et al. [47] provide specific examples where oblique views are necessary to diagnose particular conditions such as orbital masses or meningioma in the optic sheath. By construction however, most acquisition devices can only provide views in the three main orthographic directions. Therefore oblique views need to be obtained by computer reconstruction from orthographic slices. As a preliminary step, the series of evenly spaced 2D slice views produced by the acquisition device are stacked onto one another in order to create a virtual 3D view of the examined region. This 3D volume makes it possible to reconstruct slices of arbitrary direction.

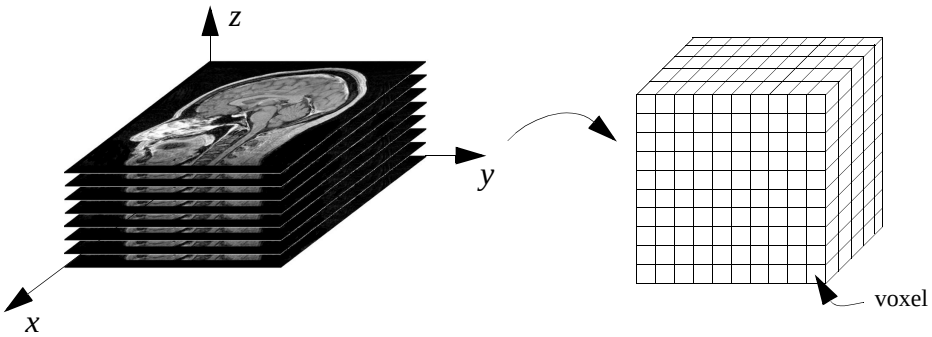


Figure 7-2: Stacking 2D slices to create a 3D volume

7.2. Algorithm overview

7.2.1. Objectives and design

The increasing size of the volumetric datasets produced by medical imaging modalities is bound to become one of the major concerns when designing medical visualization algorithms. However research in the fields of visualization has traditionally focused on raw computing performance improvements with little attention paid to storage and I/O issues. A common assumption is that the hardware has enough main memory to hold the whole image at once and that this memory is uniformly accessible (shared-memory paradigm) [51]. But these machines are out of the reach of the general practitioners and can only be afforded by important hospitals or clinics.

Taking these considerations into account, we propose here a parallel distributed slicing algorithm based on discrete geometry called *DigiPlan*. Its advantages over previous implementations are:

- high-performance thanks to parallelism at two levels: computation and disk access, which are furthermore pipelined (see Section 7.3);
- the size of the processed images is not limited by the available memory but rather by disk space;
- pure software implementation with no need for dedicated rendering hardware thereby ensuring better portability across a wide range of machines
- allows the application to run both on a single-processor PC and a network cluster of PCs.

Close integration of DigiPlan within the CAP/PS² parallelization framework enabled us to achieve this goal Such features would have been very difficult to implement, not to say impossible within the available time frame using ad hoc parallelization code. Thus, even though DigiPlan uses an abstraction layer that makes it more or less independent of the underlying image library and parallelization substrate, its current implementation is tightly bound to CAP/PS² which largely guided its design.

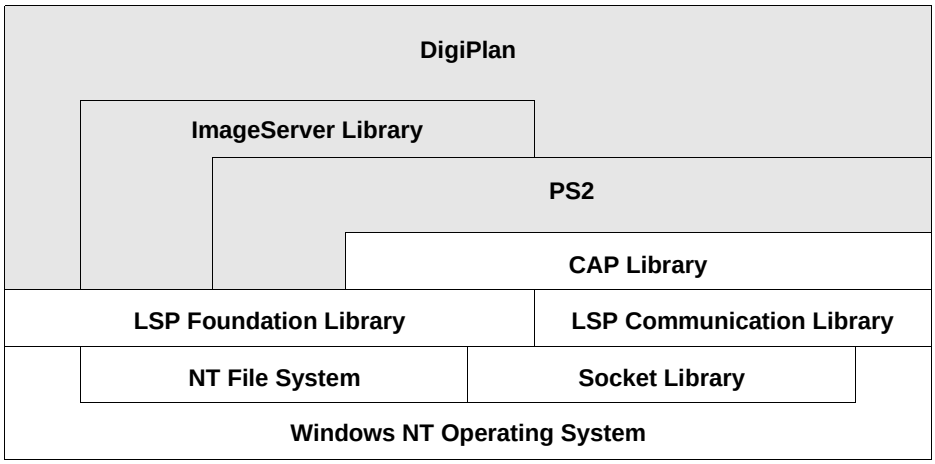


Figure 7-3: Integration of DigiPlan within the CAP/PS² framework

Figure 7-3 shows the overall design of the integration of the DigiPlan library within the CAP/PS² framework.

At the bottom, the Windows NT operating system is insulated from the above layers by two libraries: the *LSP Foundation Library* and the *LSP Communication Library* which provide respectively basic data structures, memory pool allocation routines and OS independent interprocess communication primitives.

The *CAP Library*, which stands for Computer Aided Parallelization Library, provides the necessary functions to implement the features of the CAP language, i.e. sequencing of operations, thread/process management and synchronization, inter-address-space message-passing, etc... The *PS²* system together with the CAP Library upon which it depends, constitute the core of the CAP/PS² framework.

PS² offers the necessary primitives to access striped files distributed over a cluster of networked machines each having several disks attached (see Figure 6-3). A PS² file is divided into a series of atomically addressable chunks called *extents*. PS² is perfectly general and makes no assumption on the nature of the files it handles, thus it does neither determine the size of the extents nor their distribution on the disks, both of which are application-dependent parameters. On the other hand, on each disk, it stores the extents belonging to the same striped file into a single NTFS file. It also provides open/close primitives at the striped file level and exports computation threads into which custom operations can be plugged, thus ensuring a very tight combination of disk access and computation operations [44].

The *ImageServer Library* completes PS² by offering primitives dedicated to the handling 2D and 3D images. ImageServer image files are PS² files and thus are divided into extents. 2D image extents are actually rectangular subtiles of the original image, while 3D image extents are rectangular parallelepipedic subvolumes of the original image (Figure 7-4).

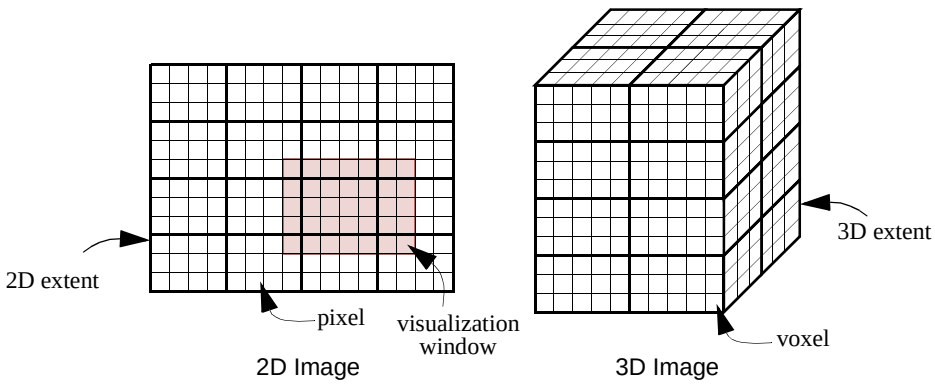


Figure 7-4: 2D and 3D Image Extent Partitioning

Considering that image access patterns to large 2D or 3D images generally consist of contiguous regions (visualization windows), extent based image access when combined with an appropriate distribution of the extents onto the disks has been shown to be very efficient [31, 26]. Indeed, storing a large 2D image scanline by scanline or a large 3D image plane by plane results in poor access times when a small contiguous portion must be visualized since much more data than necessary needs to be read from the disks. Partitioning a large image into rectangular (resp. parallelepipedic) extents greatly improves access times for this type of visualization requests.

The extent size in pixels and hence its overall size in bytes is an important factor of performance: too small an extent increases the overhead due to accessing extents and to disk latency, too large an extent increases the amount of data to be fetched to visualize a given window. In practice the extent size should be kept between 12 and 48 KBytes [31]. The extent dimensions are specified by the user at image creation time (or when images are converted to the ImageServer's format).

Extent distribution is what ultimately balances the load of the visualization system. A good distribution must ensure that for any visualization request, the affected extents are distributed onto as many disks as possible. This task is handled by the ImageServer library and is achieved by introducing, between two successive rows of extents and between two successive planes of extents, offsets which are prime to the number of disks.

Finally, the *DigiPlan* library sits on top of all these layers. It uses the ImageServer library to open/close the 3D image files and query their geometric and storage parameters (image size, extent size, number of bytes per pixel, etc...) Then it uses directly the PS^2 extent reading primitive in order to achieve maximal performance (*extent read/write* are the fundamental operations, hence the fastest I/O operations in the CAP/PS^2 framework).

7.2.2. Discrete plane scanning algorithm

Extracting oblique slices from 3D volumetric data is a problem that has focused some attention in the past and for which several methods have been proposed. The earliest as cited in [47] used a very cumbersome approach consisting in transforming the entire 3D volume according to the oblique direction to make it perpendicular to the viewer. Pixels were then removed in front of and behind the plane of interest. This, of course, requires large amounts of processing time and is very inappropriate as the computation time of such an algorithm is a linear function of the 3D volume size when it should be a function of the size of the 2D plane part to display.

Other intuitive approaches consist in direct computation of the final slice pixels using some form of floating-point interpolation (generally tri-linear) of the voxel values in the 3D dataset [45]. But tri-linear floating-point interpolations are a costly operation which makes handling large images awkward on small computer systems. Hardware implementations in the form of 3D texture mapping engines exist [25], especially from Silicon Graphics (SGI), but this class of hardware remains expensive and though a de facto standard like OpenGL ensures application portability, existing implementations perform poorly on low-end non-hardware accelerated systems.

More original approaches, such as operations in the Fourier domain have also been proposed [37]. Though conceptually interesting, the necessity of going back and forth between the image and frequency domains makes such a method computationally expensive and thereby inadequate to process large images fast.

Rhodes et al. [47] presented a particularly innovative method at the time it was developed, which tried to draw some benefits from the discrete nature of the volumetric dataset produced by X-ray CT scanners. That method could still be deemed interesting today if it was not built on so shaky theoretical foundations. Nevertheless it formed the ground for subsequent approaches based on discrete geometry, including ours. The development apparently proceeded directly from the need to avoid artifacts in the views of oblique planes and could not be rooted in discrete geometry methods that were developed later. Thus the surfaces that were extracted by that algorithm were rather coarse approximations of euclidean planes by discrete sets of voxels that do not fit the current definition of digital planes.

In order to take full advantage of the discrete nature of the 3D datasets generated by medical acquisition devices, we propose an original approach based on discrete geometry. Its advantages over previous methods reside in its high efficiency and numerical stability thanks to the use of integer arithmetic, as well as its rigorous geometrical foundation which guarantees images of good quality and makes the parallelization of the algorithm easier.

As shown in Section 2.3, rational euclidean planes have a digital 18-connected counterpart called *naive digital planes* defined by an equation of the form:

$$P(a, b, c, \gamma) = \{(x, y, z) \in \mathbb{Z}^3 / \gamma \leq ax + by + cz < \gamma + max(|a|, |b|, |c|)\} \qquad (7\text{-}1)$$

$P(a, b, c, \gamma)$ represents the digitization by the closest integer point of the euclidean plane $ax + by + cz = \gamma + \left\lfloor \frac{max(|a|, |b|, |c|)}{2} \right\rfloor$.

Now without loss of generality, let us suppose that all three coordinates a, b, c are positive and that the main direction of the normal to P is z, i.e., $c = max(a, b, c)$. Inequality 7-1 leads to the following equality :

$$z = -\left\lceil \frac{ax + by - \gamma}{c} \right\rceil \qquad (7\text{-}2)$$

which shows that a naive digital plane defines a map from $\mathbb{Z}^2$ onto $\mathbb{Z}$. The voxels belonging to the digital plane can therefore be determined with a double loop in x and y. Additionally $z(x+1, y)$ and $z(x, y+1)$ can be expressed incrementally with respect to $z(x, y)$. With the notation

$$r(x, y) = \left\{ \frac{ax + by - \gamma}{c} \right\} \qquad (7\text{-}3)$$

and thanks to the following equation

$$\left\lfloor \frac{a+\varepsilon}{b} \right\rfloor = \left\lfloor \frac{a}{b} \right\rfloor + \left\lfloor \frac{\varepsilon}{b} \right\rfloor + \left\lfloor \frac{\left\{ \frac{a}{b} \right\} + \left\{ \frac{\varepsilon}{b} \right\}}{b} \right\rfloor \tag{7-4}$$

we can evaluate $z(x+1, y)$:

$$z(x+1, y) = -\left\lfloor \frac{a(x+1) + by - \gamma}{c} \right\rfloor = z(x, y) - \left\lfloor \frac{a}{c} \right\rfloor - \left\lfloor \frac{r(x, y) + \left\{ \frac{a}{c} \right\}}{c} \right\rfloor \tag{7-5}$$

Let us distinguish two cases:

- $a < c$, Equation 7-5 becomes

$$z(x+1, y) = z(x, y) - \left\lfloor \frac{r(x, y) + a}{c} \right\rfloor \tag{7-6}$$

- $a = c$, then Equation 7-5 becomes

$$z(x+1, y) = z(x, y) - \left\lfloor \frac{a}{c} \right\rfloor = z(x, y) - 1 \tag{7-7}$$

Additionally

$$r(x+1, y) = \left\{ \frac{r(x, y) + a}{c} \right\} \tag{7-8}$$

Considering that a similar evaluation of $z(x, y+1)$ is also possible, Equations 7-6, 7-7 and 7-8 are synthesized in the algorithm of Figure 7-5. This algorithm uses integer arithmetic only and has a very tight loop core which can easily fit into the primary level cache of the processor.

7.2.3. Final backward mapping and the zooming extension

Slicing through the volume actually means extracting a rectangle, called *visualization rectangle* (Figure 7-6) from the volume and displaying it on the screen.The geometric specification of a visualization rectangle consists of five parameters defined in the absolute coordinate system (see Figure 7-13) : its *normal*, as an integer vector denoted **n**, which defines the viewing direction, the *center* of the visualization rectangle within the 3D dataset as an euclidean point (real), denoted Ω, a second integer vector called *down*, denoted **d**, orthogonal to the normal, which defines the up/down orientation of the rectangle display on the screen and finally two integers, *width* and *height* which define the size of the visualization rectangle.

```
zy = -Div(a*xmin+b*ymin-gamma,c)          Integer division
ry = Rem(a*xmin+b*ymin-gamma,c)           Integer remainder
for y=ymin to y=ymax do
   z = zy
   r = ry
   for x=xmin to x=xmax
      GetValueofVoxelAt(x,y,z)
      r = r+a
      if r>=c then
         r = r-c
         z = z-1
      endif
   endfor x
   ry = ry+b
   if ry>=c then
      ry = ry-c
      zy = zy-1
   endif
endfor y
```

Figure 7-5: Naive discrete plane scanning pseudo-code

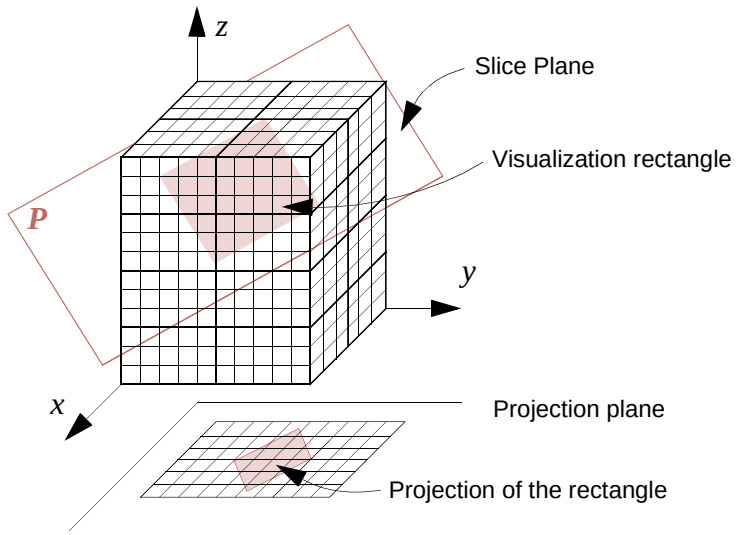
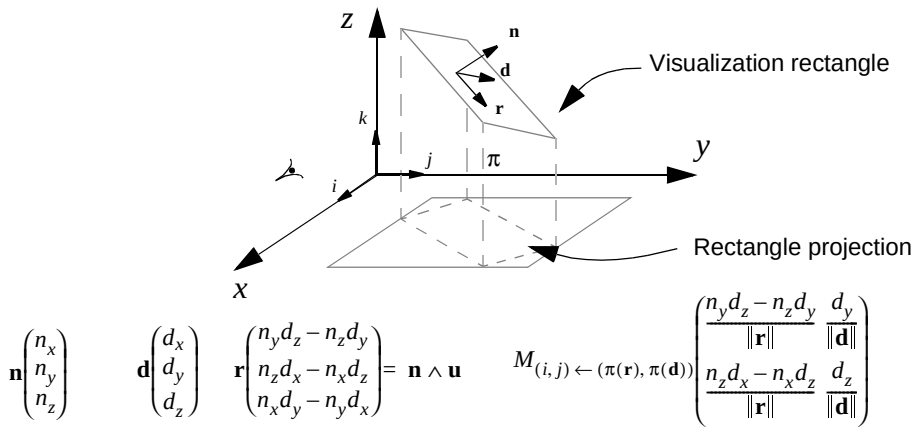


Figure 7-6: Extracting an oblique slice from a 3D voxel dataset

Scanning the plane supporting the visualization rectangle using the previous algorithm creates a 2D projection image that is not suitable for direct display on screen since selected voxels are in the original volume 3D grid. Figure 7-7 summarizes the geometry of the scene and the matrix M that maps from the screen space onto the projection space in the case the main direction of the normal, **n**, is z, i.e. $c = max(|a|, |b|, |c|)$.

A final backward mapping transforms the intermediate rectangle projection into the final screen view. It operates in a double loop on the screen space applying the matrix M to determine the coordinates of the corresponding points in the projection space. The displacements in the projection space corresponding to a unit step in the x and y directions in the screen space are



$$\mathbf{n}\begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} \qquad \mathbf{d}\begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} \qquad \mathbf{r}\begin{pmatrix} n_y d_z - n_z d_y \\ n_z d_x - n_x d_z \\ n_x d_y - n_y d_x \end{pmatrix} = \mathbf{n} \wedge \mathbf{u} \qquad M_{(i,j) \leftarrow (\pi(\mathbf{r}),\, \pi(\mathbf{d}))} \begin{pmatrix} \frac{n_y d_z - n_z d_y}{\|\mathbf{r}\|} & \frac{d_y}{\|\mathbf{d}\|} \\ \frac{n_z d_x - n_x d_z}{\|\mathbf{r}\|} & \frac{d_z}{\|\mathbf{d}\|} \end{pmatrix}$$

Figure 7-7: Geometry of the projection scene

derived from the columns of M. In general, these displacements are irrational but can be approximated by rationals with an arbitrary precision. Therefore, the double loop can be performed incrementally using integer arithmetic with no loss of precision. Note also that this step is a 2D resampling, not to be mistaken with the computationally much more intensive direct tri-linear volume resampling used by other slicing methods. Furthermore, for maximum efficiency we use nearest neighbor interpolation at this point, though bilinear interpolation could naturally also be used. In the case of very high resolution images like the Visible Human, the difference is perceptually negligible in most situations and does not justify the computation increase.

This last resampling step is not a heavy penalty, first because it is a 2D operation that can be optimized more easily than tri-linear interpolation and secondly because one can make the most of it by using it to scale the displayed image. Indeed, introducing a multiplicative coefficient in the resampling matrix coefficients provides a convenient means of applying a zoom factor, thereby allowing the display of downscaled slices at no extra cost. This is particularly useful to produce global views of very large datasets like the Visible Human (Section 7.5).

7.3. Parallelization considerations

7.3.1. Design

Now considering that the volumetric dataset is partitioned into extents distributed over several disks, a parallel version of the slicing algorithm requires the following operations: (1) determine the extents intersecting the visualization rectangle, (2) read these extents from the disks, (3) extract the digital plane part contained in each extent and resample it for screen display, (4) merge the individual resampled plane parts into the final screen display buffer.

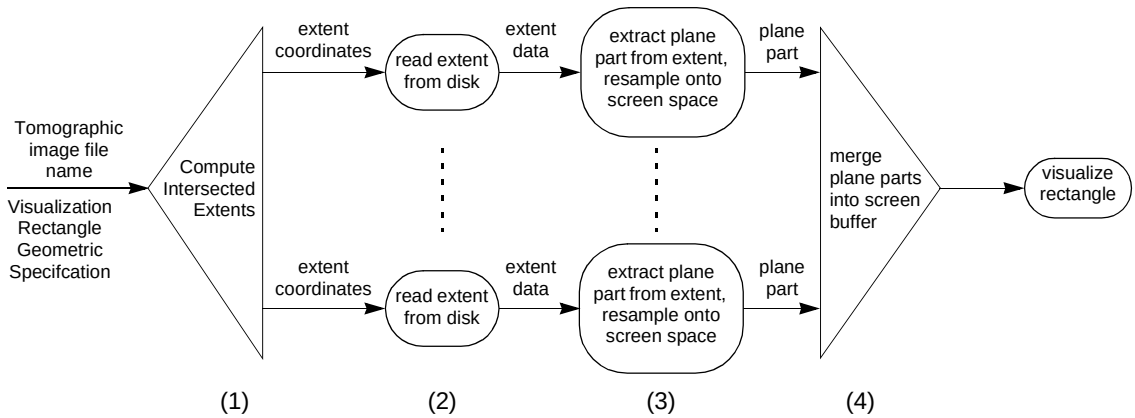


Figure 7-8: Graphical flow-chart of the parallel slicing algorithm

As illustrated in Figure 7-8, these elementary operations can be both pipelined and carried out in parallel. Pipelining can be achieved at three levels:

- a digital plane part can be extracted and resampled while the next one is being read from the disk
- a resampled plane part can be merged into the display screen buffer while the next one is being extracted and resampled
- a full visualization rectangle is displayed while the next one is being extracted, in case a request for browsing several slices into the volume is made

Parallelism can be achieved at two levels:

- several extents can be fetched simultaneously from the disks. Thus, increasing the number of disks provides an increased aggregate I/O throughput
- digital plane parts can be extracted from several extents at the same time if several processors are available. Thus, increasing the number of computation slave nodes increases the overall performance of the extraction and resampling computations.

The individual operations (1), (2), (3) and (4) are purely sequential operations, only their arrangement shown in Figure 7-8 introduces parallelism and pipelining. Transcribing such a graphical flow chart into working code is the purpose of the CAP extension to C++ which was used to implement the parallel slicing algorithm. Figure 7-9 shows the corresponding CAP pseudo-code (for the sake of clarity the syntax has actually been slightly simplified). From this very short synthetic description, the CAP preprocessor generates all the necessary code to create the appropriate processes and threads, to schedule the individual operations, and to exchange data between threads possibly located in different address-spaces (when running in a multi-PC environment).

```
int ComputeIntersectedExtents(PlaneExtractionParameters* parameters,
                              ExtentReadRequest* request)
{ // C++ code }

void MergePlaneParts(Plane* screenBuffer, PlanePart* planePart)
{ // C++ code }

leaf operation PlanePartExtraction
   in    Extent* extent
   out   PlanePart* planePart
{  // C++ code }

operation Ps2Server::PlaneExtraction
   in    PlaneExtractionParameters parameters
   out   Plane plane
{
   parallel while (ComputeIntersectedExtents,
                   MergePlaneParts, Client, Plane plane)
   (
      ExtentServer[thisTokenP->ExtentServerIndex].ReadExtent
      >->
      ComputeServer[thisTokenP->ExtentServerIndex*
                    NB_OF_NODES/NB_OF_DISKS].PlanePartExtraction
   );
}
```

Figure 7-9: CAP pseudo-code of the parallel slicing algorithm (1st variation)

7.3.2. Expected gains

The slicing algorithm is particularly well suited for parallelization, indeed there is no data dependency between the computations made on each extent. Thus each storage and processing node can work independently on its set of extents without needing to exchange information with the other nodes. This greatly simplifies the design of the parallel algorithm and reduces the amount of communications which guarantees a very good parallelization speedup. Actual performance measurements show that the speedup is linear from one to five computation nodes loaded with 12 SCSI disks (Section 7.4).

7.3.3. Possible Variations

In the proposed configuration (Figure 6-3), the network appears as the only resource which is not easily scalable. Indeed both the I/O and computation throughput can be improved by increasing respectively the number of disks attached to the slave nodes and the number of processors per node or the total number of nodes. On the other hand, increasing the network bandwidth can only be done by either switching to another technology (and there is no other technology offering more than 100 Mb/s in the same price range as Fast-Ethernet today) by changing the network topology (subnets) or by introducing an expensive high-speed crossbar switch.

Under those circumstances, particular attention must be paid to the total volume of data that travels across the network. Two variations of the parallelization have thus been considered.

1. The determination of the extents intersecting the desired slice (hit extents) is performed on the master. This simpler variation corresponds to Figure 7-9. It ensures that the hit extents are computed only once for each extracted slice. This computation is performed on the master node which sends to the slave nodes through the network the individual extent extraction and resampling requests.

2. The determination of the hit extents is performed on the slaves. In this configuration only the geometric specification of the visualization rectangle travels through the network from the master node to the slaves. Then each slave itself computes the hit extents and processes those that are locally stored. This variation minimizes the network traffic at the expense of additional computation, since the same determination of the hit extents is performed on each slave when it could have been done only once on the master.

```
int DuplicateParameters(PlaneExtractionParameters* parameters,
                        NodeLocalPlaneExtractionParameters* copy)
{ // C++ code }

int ComputeLocalIntersectedExtents( NodeLocalPlaneExtractionParameters* parameters,
                                    ExtentReadRequest* request)
{  // C++ code
   // Compute the extents that intersect the visualization rectangle and that are stored on the current node
}

void MergePlaneParts(Plane* screenBuffer, PlanePart* planePart)
{ // C++ code }

leaf operation PlanePartExtraction
   in    Extent* extent
   out   PlanePart* planePart
{  // C++ code }

operation Ps2Server::PlaneExtraction
   in    PlaneExtractionParameters parameters
   out   Plane plane
{
   indexed (int nodeIndex=0; nodeIndex<NB_OF_NODES; nodeIndex++)
   parallel (DuplicateParameters, MergePlaneParts, Client, Plane plane)
   (
      parallel while (ComputeLocalIntersectedExtents,
                      MergePlaneParts, Client, Plane plane)
      (
         ExtentServer[thisTokenP->ExtentServerIndex].ReadExtent
         >->
         ComputeServer[thisTokenP->ExtentServerIndex*
                       NB_OF_NODES/NB_OF_DISKS].PlanePartExtraction
      );
   );
}
```

Figure 7-10: CAP pseudo-code of the parallel slicing algorithm (2nd variation)

Experiments have shown that the network throughput is the first bottleneck when the number of processing nodes increases. The overall system is therefore much better balanced in the second variation.

Note that switching from one parallelization strategy to the other is just a matter of changing a few lines of the CAP parallel code as shown in Figure 7-9 and Figure 7-10.

7.3.4. Determination of the hit extents

The determination of the extents hit by the visualization rectangle uses results in discrete geometry that were developed in the first half of this work, namely the covering of a digital parallelogram by a low resolution grid (Section 5.3). Indeed the projection of the extent grid on the projection plane forms a tessellation of that plane by rectangular tiles. The algorithm of Section 5.3 allows to find very efficiently the tiles that cover the projection of the visualization rectangle which is a parallelogram in the general case. The set of tiles that intersect the parallelogram is nothing more than the projection of the extents hit by the visualization rectangle. (Figure 7-11).

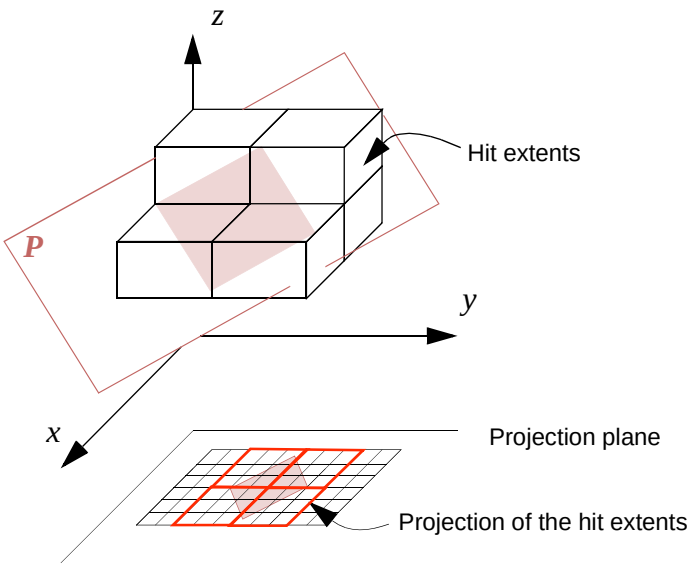


Figure 7-11: Determining the extents hit by the visualization rectangle

Knowing the projection of the hit extents and the equation of the plane supporting the visualization rectangle, one can determine the elevations of the hit extents. Depending on the extent dimensions, it is possible for the extraction rectangle to intersect two or more vertically adjacent extents. Therefore, for each projection of an intersected extent, the algorithm evaluates the minimum and maximum elevations of the slicing plane above. Since the plane is a linear geometric object, the process can be optimized by noticing that the extremal values of the elevation are reached above two of the four corners of the extent projection and that those corners are the same for all the extent projections.

Let us consider a visualization rectangle lying on a plane of normal $\mathbf{n} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$ where $c > 0$ is the biggest coordinate in absolute value. In this case the projection plane is the plane containing the x and y axes. Figure 7-12 summarizes where the corners of minimum/maximum elevation are situated on the extent projections according to the values of a and b.

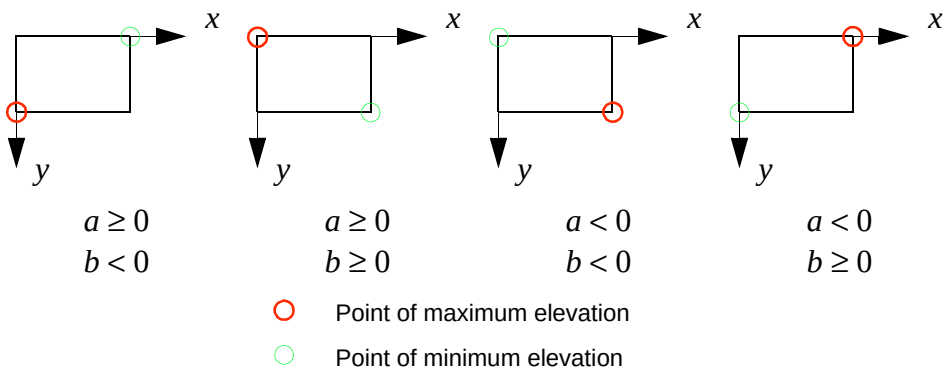


Figure 7-12: Corners of max/min elevation on the extent grid projection

Moreover, much in the same way as elevations of adjacent voxels can be computed incrementally in the naive digital plane scanning algorithm, the min/max elevations on extent projection boundaries can also be calculated incrementally. Therefore the determination of the hit extents is a pure discrete problem solved with a pure integer arithmetic algorithm.

7.3.5. Non-isometric volumes

Due to technical constraints, biomedical 3D images often are anisotropic, i.e., the resolution of the acquisition along the three axes is not identical. Thus the pixel size on each of the 2D slices making up the 3D dataset may be smaller than the distance between two consecutive slices. For instance, the Visible Human Male dataset scanned by the U.S. National Library of Medicine, Bethesda, has a resolution of one third of a millimeter on each slice while the slices are spaced at a one millimeter interval.

This is undesirable since most visualization algorithms, including ours, require the voxels to have the same size along the three main directions. Therefore medical image visualization often includes a preprocessing step of the whole 3D dataset consisting in the interpolation of the missing voxels in order to build an isotropic dataset having the same resolution along the three main axes [52]. Sophisticated methods for interpolating the missing slices have been developed. For instance Ruprecht and Müller proposed a method inspired from morphing techniques where each slice is deformed into the next one [49]. However if the slice/inter-slice resolution ratio is

not too high, a computationally less expensive linear interpolation between the slices gives good results and is therefore often preferred.

However doing the interpolation during a preprocessing step, though it ensures each voxel is interpolated only once, results in an artificial image size increase that causes lower overall performance of the visualization system as more data needs to be stored and read from the disks. This is especially true for algorithms like the proposed digital plane scanning algorithm which ensures each interpolated voxel is accessed at most once.

Therefore we propose a solution where voxels in missing slices are resampled on the fly. This is made possible by using a double set of coordinates. *Physical coordinates* relate to the original 3D dataset while *virtual* (or *absolute*) *coordinates* relate to an abstract object representing the volume as if it had been previously appropriately resampled (Figure 7-13). The digital plane

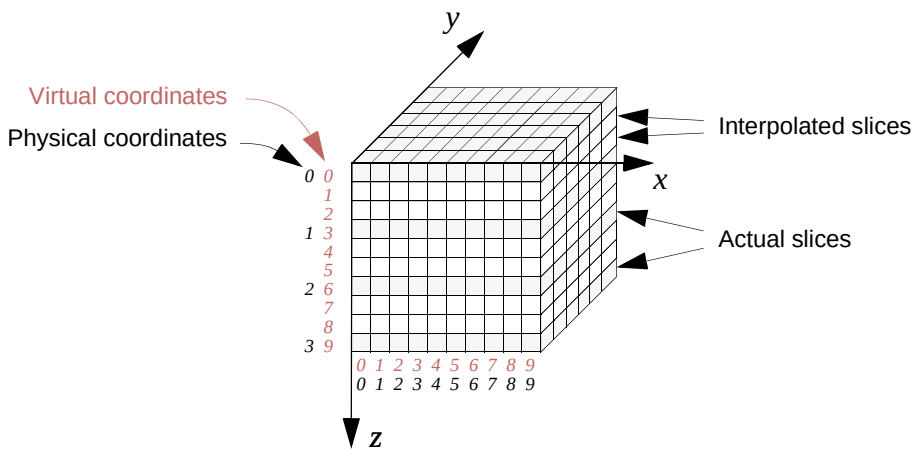


Figure 7-13: Physical and virtual coordinate systems

A sample 3D dataset where the in-slice resolution is three times as high as the inter-slice resolution.

scanning algorithm operates in the virtual coordinate system in the exact same way as if the whole dataset had been previously resampled. Then the pseudo-function `GetValueOfVoxelAt` (Figure 7-5) is simply extended so as to interpolate voxels that do not actually exist in the volume. For instance, consider the function is asked for the value of voxel $(3, 0, 7)$ in the volume of Figure 7-13, it simply returns a value interpolated from the voxels of physical coordinates $(3, 0, 2)$ and $(3, 0, 3)$.

One additional difficulty arises from partitioning the volume into extents. In some situations, the voxels needed to interpolate a virtual voxel may reside in different extents. This situation creates a prejudicial dependency on neighboring extents which can affect overall performance. Thus in a worst case situation, two layers of extents could be read to recreate a missing slice resulting in twice as much data as necessary being read from the disks. To solve this problem,

we introduce redundancy in the storage format through the creation of so called *fat extents*. A fat extent contains as a last layer in the undersampled direction, an additional layer of voxels which is copy of the first layer of the next adjacent extent in that direction. Figure 7-14 shows

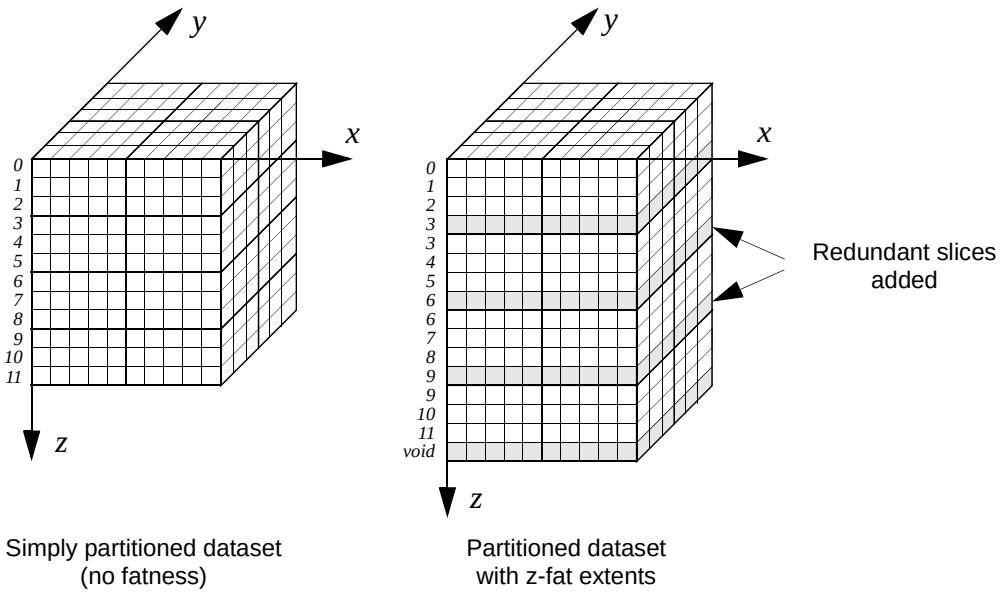


Figure 7-14: Volume partitioning using fat extents

how this principle would be used for a volume needing reconstruction of slices along the z-axis. If resampling is needed in more than one direction, the mechanism can be extended and fatness layers can be added for up to the three directions x, y and z.

The volume size increase thus created is relatively small. For instance if the extent size in the z direction is 32, then using z-fat extents makes the volume 1/32nd bigger.

7.4. Measured performances

In this section we provide the results of experimental performance measurements made on the DigiPlan parallel slice extraction application. Naturally, the performance of the overall system largely depends on the performances of the underlying PS^2/CAP parallelization framework and on the fine-tuning of the parallelization strategy with respect to the available hardware. Detailed results from this viewpoint can be found in [43] and [44]. Performance figures in this section are courtesy of Vincent Messerli [44].

The measurements were made on a parallel system consisting of a network cluster of 200 MHz Bi-PentiumPro PCs, 1 client (master) + 1 to 5 server nodes (slaves), connected by a Fast Ethernet network at 100 Mbits/s. Each of the slave PCs was loaded with up to 4 SCSI-2 strings (3 disks per string) each offering a maximum nominal throughput of 10MBytes/second. The

experiments consisted in extracting 512x512 24 bit color slices from the Visible Human male dataset (13 GB).

Figure 7-15 shows that the system scales linearly as well with the number of disks as with the number of server nodes.

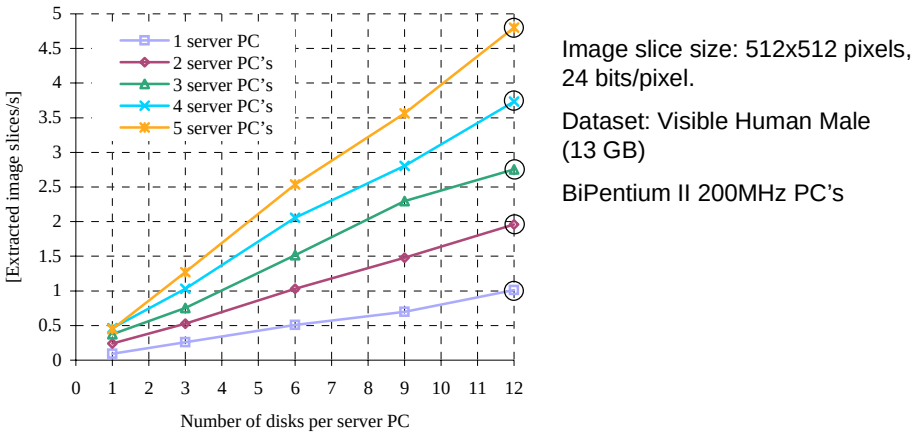


Figure 7-15: Scalability across the number of slave PCs and disks

The system reaches its peak performance at 4.8 512x512 color images per second on a configuration consisting of five server nodes and one client. At this point the processors of the server nodes are loaded at 80% while the client node processor is loaded at nearly 85% (Figure 7-16). The weak spot of the system seems therefore to reside in the limited processing

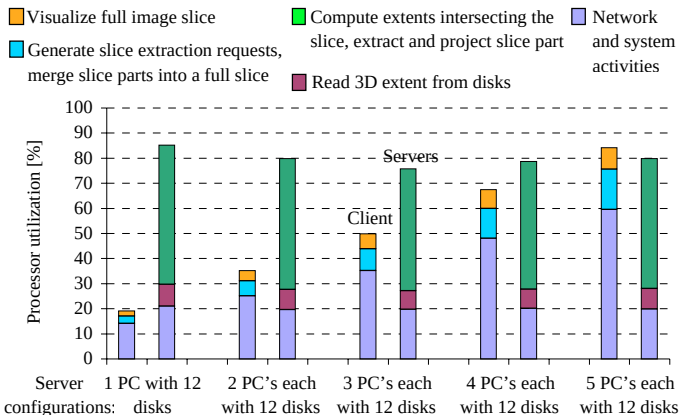


Figure 7-16: Master and slave processor utilization

power of the client node which could be attributed to the extraction algorithm, or more precisely to the merging of the received plane parts into the final display buffer. Actually Figure 7-16 shows that the master node spends 60% of its processing time on network operations and less than 15% on computation. In fact the bottleneck resides in the limited throughput of the client

node's interface to the network. Therefore the most efficient optimization would not come from an improvement of the merging algorithm but rather from an intelligent network adapter which could off-load network protocol related operations off the master node's main CPU (I^2O technology).

7.5. The Visible Human Slice Server

The most spectacular application of the DigiPlan library for the extraction of digital planes from 3D voxel-based volumes is certainly *The Visible Human Slice Server* [54]. This web site (`http://visiblehuman.epfl.ch/`) which was unveiled to the public in June 1998 has served nearly 140'000 slice extraction requests in less than 5 months and has been lauded by the press and the Internet community.

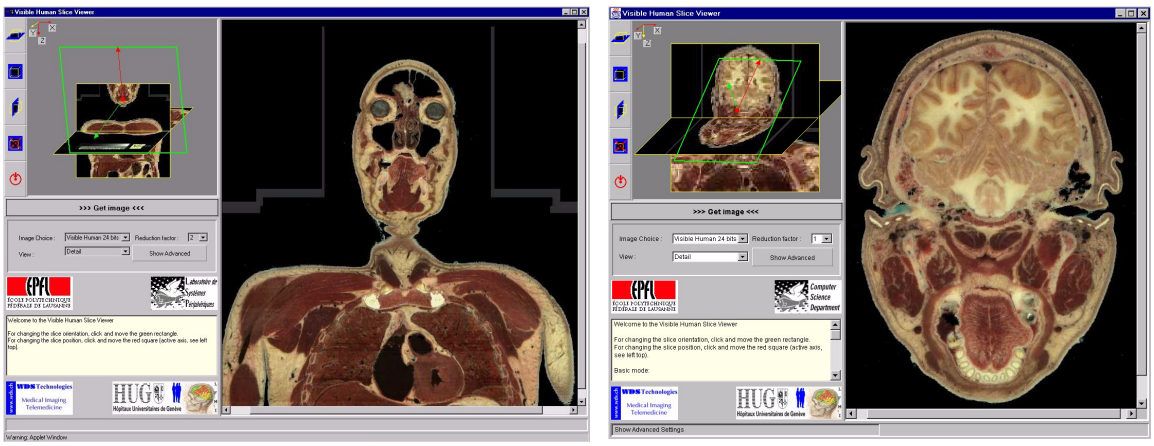


Figure 7-17: The Visible Human Slice Viewer Java applet

Thanks to this application, the public is able to extract interactively high-resolution oblique slices of arbitrary orientation from within the Visible Human male dataset. This unprecedented high-quality volume consists of axial cryosection full-color photographs of a complete human body from head to toe. Each axial section has a pixel resolution of one third of a millimeter, the inter-slice resolution being 1 mm. The total amount of data represents 13 GB. A smaller volume comprising only the head of the visible human (1 GB) and an MRI dataset (100 MB) are also available on-line for browsing. A similar dataset of a female body will also be available in the near future.

When a visitor accesses the web site, a Java applet is downloaded to his World Wide Web browser. This applet allows him/her to choose a 3D volume to browse, to specify the parameters of the slice to extract (orientation, size) either interactively using the mouse or by filling input text fields. The user can then ask for the corresponding slice from the server. The request is sent to the server by the Java applet through the Internet. The server extracts the corresponding slice using the DigiPlan algorithm presented throughout the previous sections, compresses it using a

JPEG compression scheme (for faster transfer over the Internet) and sends it back to the Java application on the client for display.

The experimental configuration of the Web parallel server extends the traditional CAP/PS^2 hardware architecture (Figure 6-3) by adding a web server (HTTP server) process on the client node which thus becomes the interface between the local network cluster where the slice extractions are performed and the Internet (Figure 7-18).

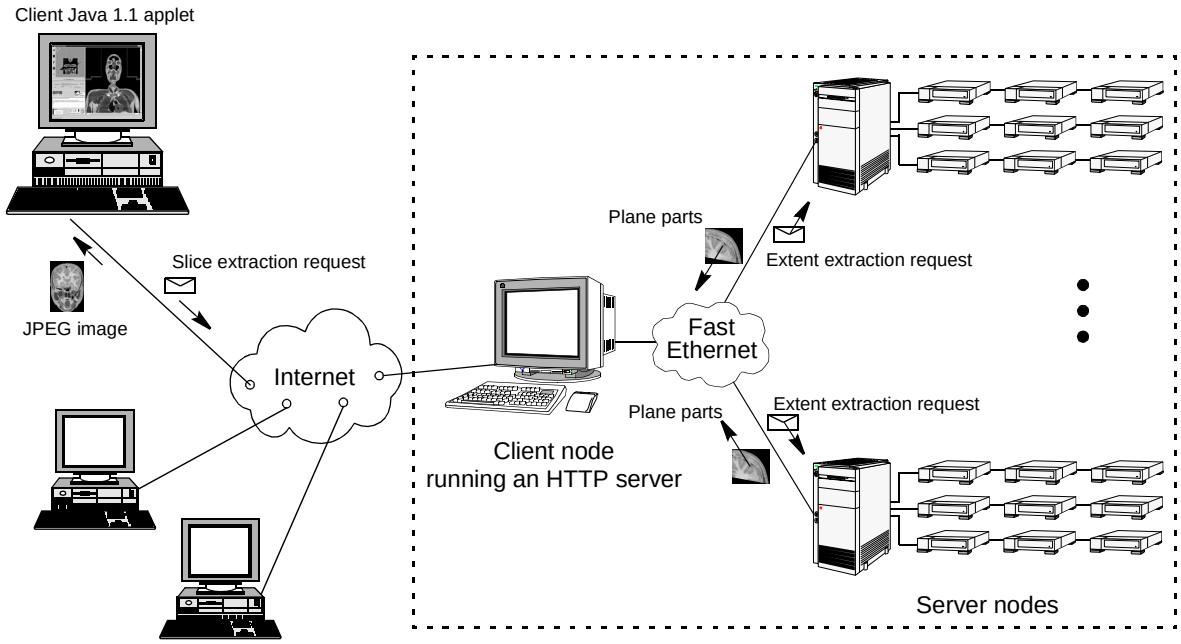


Figure 7-18: Architecture of the Visible Human Web Server

The configuration currently in production as a web server, actually consists of a single dual-Pentium II 300MHz PC with 16 disks attached onto which the Visible Human dataset is striped. Oblique slices are extracted and compressed in nearly one second, which given the current bandwidth of the Internet can be considered as an interactive rate. The overall capacity of the system could thus theoretically reach 3’000 extractions per hour in this configuration. Should an higher rate be desired, the system could be easily scaled up by adding slave storage and computation nodes with no need for an application rewrite.

7.6. Summary

We have presented in this section an innovative approach to extract planar slices of arbitrary orientation from a 3D discrete volume. Such an application is a fundamental tool in biomedical imaging, where volumes produced by modalities sucha as X-ray CT, MRI or PET are used for teaching purposes as well as for diagnosis. While previous methods either used floating-point arithmetic, expensive dedicated hardware or attempted to use discrete properties of the volume

in an awkward manner, our approach is purely software-based and relies on rigorous results of discrete geometry to draw maximum advantage of the discrete nature of the volumetric data. The key features of the proposed algorithm are:

- The oblique slice is extracted with a discrete plane scanning algorithm using integer arithmetic.
- The algorithm only needs a 2D resampling step which can be performed incrementally using integer arithmetic and a nearest neighbor interpolation. This resampling step can be used to implement zooming at no extra cost
- The determination of the volumic extents intersected by the slice uses the result established in Section 5.3 about the covering of a digital parallelogram by a regular rectangular plane tesselation. The hit extents are therefore determined exactly, in linear time using integer arithmetic.
- Non-isometrically sampled volumes are resampled on the fly. The notion of fat extents avoids all data dependencies in this case: interpolating a missing voxel never needs data from two different extents.
- The algorithm was shown to be well suited for parallelization and was integrated into the CAP/PS^2 parallelization framework where it showed good performance, allowing the overall system to scale linearly up to 6 PCs working in parallel.

The Visible Human Slice Web Server further demonstrates the qualities of the extraction algorithm and the underlying parallelization framework: scalability, efficiency and robustness. Another notable aspect of this algorithm is its generality. Though we emphasized biomedical imaging, such a parallel digital slicing algorithm could find many other application fields like physics, meteorology or geology to name a few.

8 Extraction of Digital Generalized Cylinders

This chapter presents an algorithm for extracting ruled surfaces out of 3D discrete volumes such as those produced by medical tomographic imaging devices. The algorithm based on discrete geometry can be parallelized to enable working with very large images. It offers a valuable extension to the simpler planar slicing algorithm presented in the previous chapter.

8.1. Introduction

As explained in the previous sections, most physicians still feel more comfortable today with the observation of two-dimensional views extracted from 3D volumetric datasets acquired by biomedical imaging modalities such as CT, MRI, PET, etc... Having built a habit of observing radiological images with restricted orientations during their education, it is often difficult for them to extract valuable medical hints from three-dimensional representations. However the traditional axial, sagittal, and coronal image slicing orientations are more and more considered as too much of a limitation. Extracting oblique slices of arbitrary orientation is considered as an important improvement (Chapter 7).

In some situations though, oblique plane slices are still not enough. For instance, slicing through the middle of several vertebrae with a plane is not possible because of the natural curvature of the vertebral column especially in regions of high curvature like the lumbar vertebrae (Figure 8-1). The jaw is another region where plane slicing is too limited to get a full view of

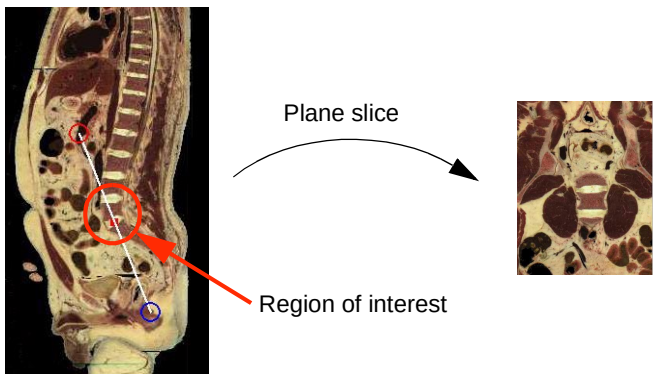


Figure 8-1: Limitations of oblique plane slicing

Of the three vertebrae in the region of interest, only one is visible in its entirety on the plane slice.

the teeth from left to right. Such views would nevertheless be very useful for dentists to design orthodontic devices. Without having recourse to three-dimensional visualization techniques, the

solution to this kind of problem resides in an extension of the usual slicing algorithms to more complex surfaces.

8.1.1. Ruled surfaces and cylinders

A class of surfaces called *ruled surfaces* is of particular interest. Differential geometry defines a ruled surface $\sigma(t, v)$ as a differentiable map defined by two elements: a curve $\alpha(t)$ of $\mathbb{R}^3$ and a parameterized family of directions $\mathbf{w}(t)$ of $\mathbb{R}^3$:

$$\sigma(t, v) = \alpha(t) + v\mathbf{w}(t) \qquad t, v \in I \tag{8-1}$$

The curve $\alpha(t)$ is called the *directrix* of the surface while the family of lines L_t passing through $\alpha(t)$ and parallel to $\mathbf{w}(t)$ are called *rulings* of the surface. Ruled surfaces having a constant tangent plane along each ruling are called *developable surfaces*. Those surfaces can be "unfolded" and "flattened" with no deformation, which is a fundamental characteristic in our case, since we want to avoid three-dimensional visualization. The most common types of developable surfaces are cones and cylinders [7] (Figure 8-2).

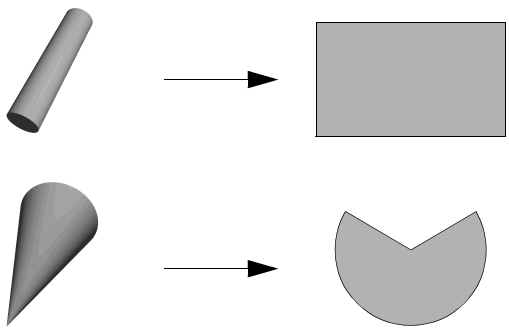


Figure 8-2: Development of cones and cylinders

We shall focus on *cylinders*. A cylinder is a ruled surface whose directrix is contained in a plane and whose rulings are of constant direction. Note that this notion encompasses but is not restricted to the usual circle-based cylinders (Figure 8-3).

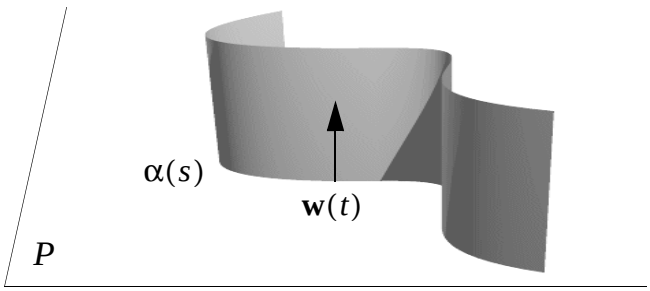


Figure 8-3: A generalized cylinder

Cylinders whose rulings are normal to the plane of the directrix are particularly well suited to two-dimensional viewing. Let us consider such a cylinder defined by the differentiable map $\sigma(s, v)$:

$$\sigma(s, v) = \alpha(s) + v\mathbf{w} \qquad s, v \in I \tag{8-2}$$

verifying the following additional conditions:

$$\begin{aligned} &\alpha'(s) \perp \mathbf{w} \qquad s \in I \\ &\|\mathbf{w}\| = 1 \end{aligned} \tag{8-3}$$

where α is parameterized by its arc length s.

The total differential of $\sigma(s, v)$ writes:

$$d\sigma(s, v) = \frac{\partial \sigma}{\partial s} ds + \frac{\partial v}{\partial v} dv = \alpha'(s) ds + \mathbf{w} dv \tag{8-4}$$

Since α is parameterized by the curve length s then $\|\alpha'(s)\| = 1$. Together with the additional conditions of Equation 8-3, this indicates that the map σ preserves lengths locally and introduces no deformation which is a key point for the visualization of cylinders on a flat display.

Also, specifying a cylinder verifying Equation 8-3 only requires determining the plane containing the directrix and then the directrix itself as a 2D curve on this plane. This makes the interaction with the user very simple with no need for a three-dimensional user-interface or pointing device.

8.1.2. Digital cylinders

In Section 2.1.3, we recalled that the definition of discrete surfaces is a complex matter and that several approaches exist. In Chapter 7, we have seen how naive digital planes were particularly well suited for slicing through 3D discrete volumes thanks to their appropriate connectivity and ease of scanning. It is therefore natural to look for similar properties in what we will define to be digital cylinders. Interestingly enough, some research has been done in the field of the incremental recognition of digital planes on discrete surfaces (discrete polyhedrization) [15]. In the same way as a digital curve is equivalent to a discrete polygonal line, i.e., a sequence of digital straight line segments, a thin digital surface can be conveniently represented as a juxtaposition of digital plane patches. Chapter 4 exploits this idea to build digital representations of Bézier curves and surfaces. Digitization of cylinders can follow the same principles, and in fact, the properties of cylinders as continuous surfaces even eliminate some of the problems mentioned in Chapter 4, namely the normal inconsistencies between adjacent planar patches.

Thus we build a naive digital cylinder by first polygonalizing its directrix $\alpha(s)$ on the directrix plane P using the algorithm of Chapter 4 (for the ease of user interaction, we restrict the directrices to natural splines, equivalent to Bézier curves). The new cylinder defined by that polygonalized directrix, denoted $\pi_\alpha(s)$, on plane P consists of a series of adjacent planar facets. (Figure 8-4). The discrete equivalent to this surface is the union of a set of digital plane

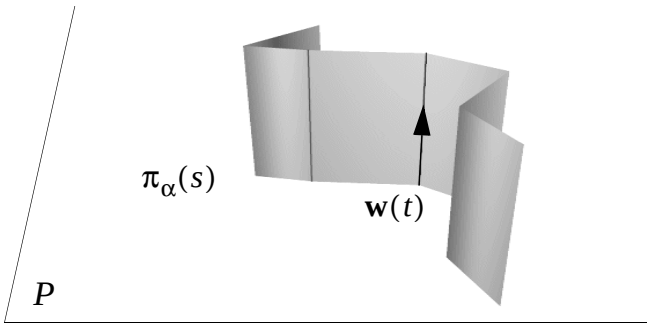


Figure 8-4: Polygonal cylinder

patches each fitting one of the facets (in the sense of a best approximation of an euclidean plane by a naive digital plane).

Appropriate connectivity at the edges can be guaranteed following ideas developed in Section 4.5.2. Each digital plane fitting one of the facets of the polygonal cylinder is defined as the set of integer points contained between two euclidean boundary planes, one on the left of the facet and one on the right[1]. The intersections of consecutive euclidean boundary planes on the same side of the cylinder make up the edges of a boundary cylinder, we call digital cylinder the set of integer points contained between the two euclidean boundary cylinders (Figure 8-5). This set of points is 18-connected.

8.2. Algorithm

8.2.1. Overall Design

From the user point of view, the interaction proceeds in three steps (see Figure 8-6):

1. Specification of the plane containing the directrix (*directrix plane*). The necessary parameters (normal, center, up direction, width and height) can be specified either interactively or by means of text input fields, just like in the Visible Human Slice Server user interface (see Section 7.5)
2. Specification of the directrix. In theory, the directrix is an arbitrary planar curve. On computer systems, common usage has shown that parametric curves of degree three

1. The parametrization of the directrix induces an orientation of the curve and hence a notion of left and right given a normal to the directrix plane.

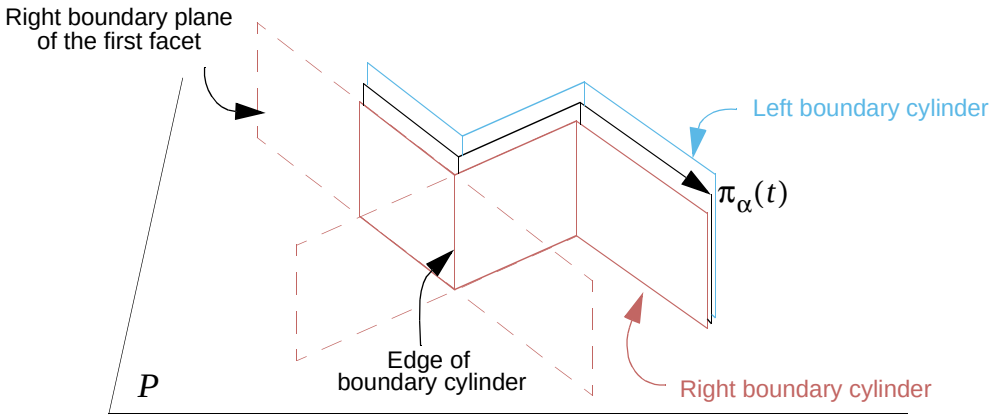


Figure 8-5: Digital cylinder boundaries

(splines or Bézier curves) provide a great flexibility and can be manipulated intuitively [23]. Therefore we restrict the directrix to this type of curves. The user specifies a series of interpolation points on the directrix plane and the natural spline that interpolates these points is drawn. Further control of the shape of the curve is then possible by moving the points or manipulating the tangents to the curve at the arc junctions. The surface width, i.e., the length the cylinder will span on each side of the directrix plane (interval of the parameter v in Equation 8-2) is also specified at this point.

3. Visualization of the extracted surface

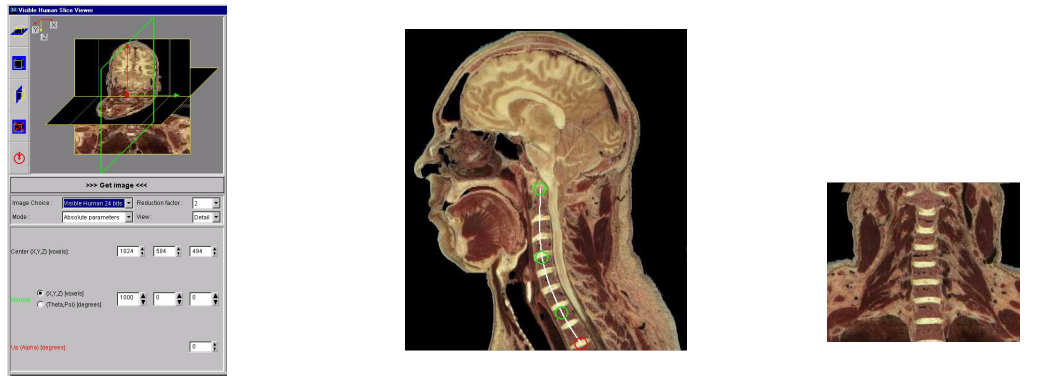


Figure 8-6: User interaction for the extraction of cylinders

Internally, the extraction of the cylinder for visualization proceeds as follows. In a first step, the directrix is polygonalized. The coordinates of the vertices of that polygonal line in the three-dimensional coordinate system of the volume are computed and this defines a digital polyhedral surface (digital cylinder) that fits the cylinder. Then the facets of the digital cylinder are

extracted as digital rectangles (using the DigiPlan algorithm, Chapter 7). Finally the 2D individual facets are merged in the final display buffer. (Figure 8-7).

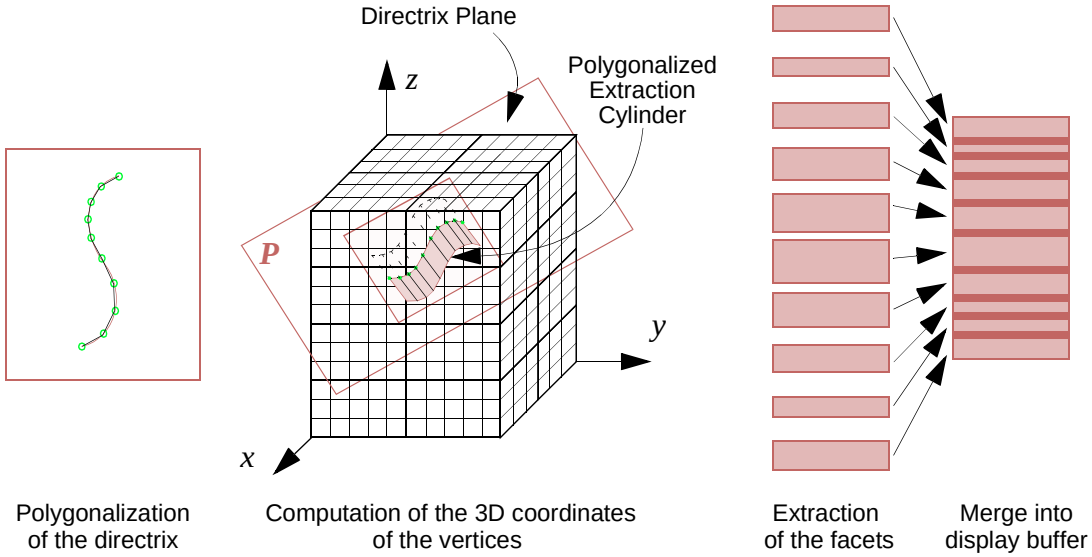


Figure 8-7: Overview of the surface extraction algorithm

8.2.2. Mapping from the directrix plane coordinates to the volume coordinates

The proposed surface extraction algorithm involves two different coordinates systems. The first one, called *virtual* (or *absolute*) *coordinate system*, denoted $(O, \mathbf{i}, \mathbf{j}, \mathbf{k})$ is related to the 3D volumetric dataset (isometrically resampled if necessary, see Section 7.3.5). The second one, called *directrix plane coordinate system*, denoted $(R, \mathbf{r}, \mathbf{d})$ is related to the rectangle upon which the user specifies the directrix (Figure 8-8). Interpolation points given by the user to define the directrix are expressed by means of pairs (X, Y) of coordinates expressed in $(R, \mathbf{r}, \mathbf{d})$. The coordinates (x, y, z) of these points in the absolute coordinate system can be computed by the following elementary relation:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r_x & d_x & R_x \\ r_y & d_y & R_y \\ r_z & d_z & R_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \qquad (8\text{-}5)$$

where (r_x, r_y, r_z), (d_x, d_y, d_z) and (R_x, R_y, R_z) represent the respective absolute coordinates of $\mathbf{r}$, $\mathbf{d}$ and R (Figure 8-8).

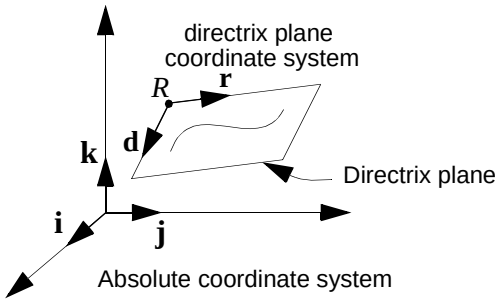


Figure 8-8: Coordinate systems for the extraction of surfaces

8.2.3. Weighting at facets joints

Section 8.1.2 defined 18-connected digital cylinders. In fact, appropriate discrete connectivity is not enough to display the surface on the screen grid. Indeed, in the general case, a discrete cylinder does not have the functionality property of naive digital planes, i.e., there is no trivial map between the cylinder and a connected subset of $\mathbb{Z}^2$. Therefore the display of the surface becomes more intricate. The key point for accurate display of the surface is the preservation of the distances so as to avoid deformations. However, arc lengths of the polygonalized directrix are irrational in the general case, whereas facets extracted as discrete rectangles have integer height. Consequently the pixel grids of the extracted facets do not match the pixel grid of the final display buffer and the facets need to be resampled into that final buffer. This situation is illustrated in Figure 8-9 for a digital cylinder consisting of two facets. One can see that due to

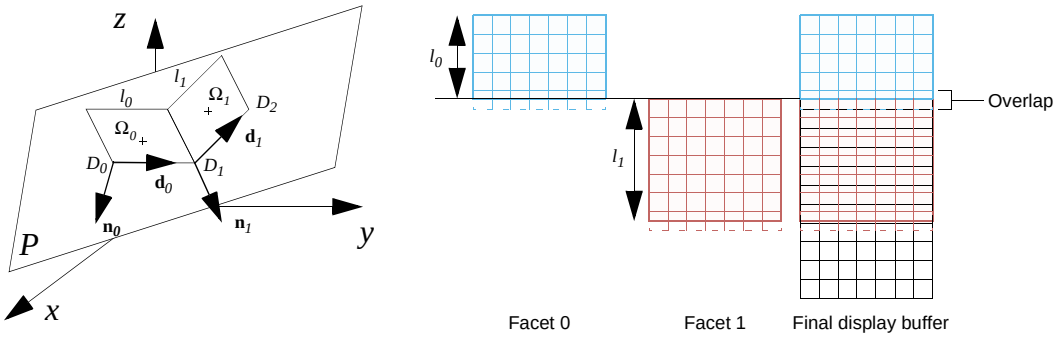


Figure 8-9: Merging facets into the final display buffer

$l_0 = \overline{D_0 D_1}$ not being an integer, the grid of facet 1 must be shifted with respect to the grid of the final display buffer, i.e., facet 1 needs resampling when merged into the final surface display buffer.

This final resampling step, though somewhat optimizable since resampling weights are constant for a given facet, represents an additional computational cost to the surface extraction algorithm. Fortunately by rethinking the facet extraction, this overhead can be avoided.

Section 7.2.3 introduces the resampling operation that maps the intermediate projection of a discrete rectangle into the viewing space. This resampling allows to extract rectangles centered on non-integer points of the discrete lattice of the 3D volume. This characteristic can be conveniently used to avoid resampling the facets into the display buffer. Indeed by shifting the center of the facets and incrementing their height appropriately we can align the facet grids with the screen buffer grid. Figure 8-9 shows that the needed center shift is essentially vertical with respect to the final display grid, which means that the center of each facet Ω_i needs to be shifted by a certain $\alpha_i \mathbf{d}_i$ where $\alpha_i \in \mathbb{R}$. Figure 8-10 which represents the scene in projection on the directrix plane P, shows how this shift $\alpha_i = \overline{D_i F_i}$ is calculated (for a directrix consisting of three edges).

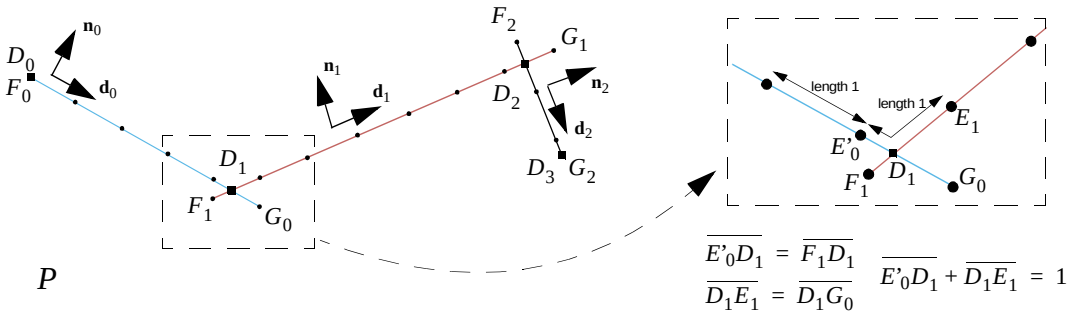


Figure 8-10: Facet shifts and height increase (projection on the directrix plane)

Small dots represent the horizontal pixel boundaries of the screen display grid. The directrix vertices are denoted D_i. The facets vectors $\mathbf{d}_i$ are defined by the relation

$$\mathbf{d}_i = \frac{\mathbf{D}_{i+1} - \mathbf{D}_i}{\left\| \mathbf{D}_{i+1} - \mathbf{D}_i \right\|} \tag{8-6}$$

The length and position of the facets that are extracted are not defined by the segments $\overline{D_i D_{i+1}}$ but rather by $\overline{F_i G_i}$ where F_i and G_i are defined by the following recurrence:

$$\begin{cases} F_0 = D_0 \\ \overline{F_0 G_0} = -[-\overline{D_0 D_1}] & (\text{roundup } \overline{D_0 D_1}) \\ \overline{F_i D_i} = \overline{F_{i-1} D_i} - [\overline{F_{i-1} D_i}] \\ \overline{D_{i+1} G_i} = -[-\overline{F_i D_{i+1}}] & (\text{roundup } \overline{F_i D_{i+1}}) \end{cases} \tag{8-7}$$

F_i is calculated so as to add an offset that compensates exactly the non-integer length of the $i-1$ facet, ensuring that the grid of facet i is well aligned on the screen grid. G_i simply rounds the facet height to the smallest greater integer.

By proceeding this way, facets extracted as discrete rectangles can be mapped directly into the surface display buffer with no additional resampling of the whole facet. However Figure 8-9 and Figure 8-10 indicate that adjacent facets may superimpose along a pixel line at their junction. This is the case for instance on the fifth pixel line of Figure 8-9 and Figure 8-10. At these junction lines the two adjacent facets must be blended together into the surface buffer. The contribution of each facet to the junction line is simply determined by the geometry of Figure 8-10. That is, the blending weights of facets i and $i+1$ at their junction is given by:

$$\begin{aligned} w_i(i) &= \overline{F_{i+1}D_{i+1}} \\ w_i(i+1) &= 1 - w_i(i) \end{aligned} \tag{8-8}$$

8.3. Parallel implementation

The extraction of digital cylinders can be performed in parallel, following the same ideas as the extraction of oblique naive digital planes. Again, the 3D volumetric dataset is physically divided into small parallelepipedic subvolumes called *extents* that are distributed among the individual disks connected to the system. The flow-chart of Figure 8-11 describes how the individual operations consistuting the surface extraction can be both pipelined and carried out in parallel.

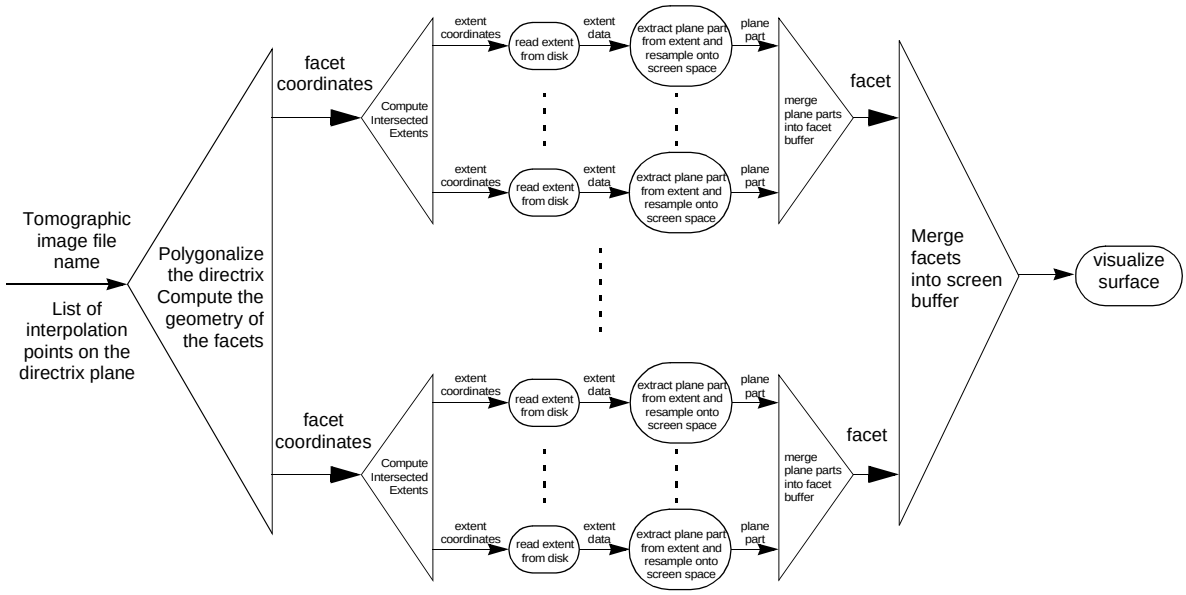


Figure 8-11: Pipelined/parallel surface extraction flow-chart

Pipelining occurs at four levels:

- a plane part can be extracted from an extent and be resampled while the next extent is read from the disk

- a plane part can be merged into a facet buffer while the next plane part is being extracted
- a facet can be merged into the screen buffer while the next facet is being extracted on the same storage and processing node
- finally, a surface can be visualized while the next one is being extracted in case a request for browsing through the volume was made

Parallelism can be achieved at two levels:

- several extents can be fetched simultaneously from the disks to increase I/O throughput
- digital plane parts can be extracted from several extents at the same time if several processors are available.

The parallel surface extraction algorithm shares most of the characteristics of the digital plane extraction algorithm that resides at its core. The extraction of plane parts from each extent still does not present any data dependency with the other extents allowing all the slave computation nodes to perform the extraction in parallel with no need for synchronization or data exchange. The only communications take place between the client node (master) and the computation server nodes (slaves) when the extraction request is sent to the slaves and the results are merged back.

Figure 8-11 shows that the flow-chart for surface extraction embeds the flow-chart of plane extraction. The CAP language allows such compositionality of operations and lets the programmer define pipelined/parallel operations including other lower-level pipelined/parallel operations. The actual implementation can thus be made very simple (Figure 8-12).

8.3.1. Possible variations

The parallelization strategy proposed in the previous section actually hides a subtile pitfall that might compromise the performance of the algorithm. Indeed a potential problem lies in the fact that a given extent may be intersected by several of the planes making up the facets of the surface. In a usual configuration, each extent lying close to a facet joint may intersect the two facets meeting at the joint, but in more unusual configurations where the extents are big and/or the facets are particularly narrow (which happens when the curvature of the surface is important) each extent may be intersected by a significant number of planes.

In the previous parallelization design, the extraction request is first splitted into a set of extraction requests for planes and then each individual plane extraction request is itself divided into volumic extent reading and processing requests. This implies that the same extent may be requested and read from the disk several times for different planes which is clearly not optimal. In fact, things are not so bad however because the PS^2 parallel file system upon which the application relies, includes a built-in cache system. Extents read from the disk are stored and kept in

```
int ComputeSurfaceFacets (SurfaceParameters* parameters,
                          PlaneExtractionParameters* request)
{ // C++ code }

void MergeFacets(SurfaceView* screenBuffer, Plane* facet)
{ // C++ code }

operationPs2Server::PlaneExtraction
   in    PlaneExtractionParameters* parameters
   out   Plane* plane
{ // Defined in Figure 7-9 }

operationPs2Server::SurfaceExtraction
   in    SurfaceParameters* parameters
   out   SurfaceView* surface
{
   parallel while (ComputeSurfaceFacets,
                   MergeFacets, Client, SurfaceView* surface)
   (
      PlaneExtraction
   );
}
```

Figure 8-12: CAP pseudo-code for the extraction of surfaces

memory using a “least recently used” cache mechanism[1]. A reasonable cache size ensures that no extent needs to be read from the disks more than once. Nonetheless this hides the design flaw only partially since fetching an extent from the cache implies a small additional overhead that can be avoided with a better parallelization strategy.

An alternative to the previous design consists in first computing all the extents intersected by the surface and grouping the plane extraction requests on a per extent basis. Then, for each extent, the plane parts corresponding to each plane it intersects are extracted. With this method each extent needs to be fetched only once, thus avoiding any overhead due to multiple extent fetches even if a cache mechanism is available. Figure 8-13 shows the flow-chart corresponding to this parallelization strategy.

The computation of the extents hit by the surface is made facet by facet. For each intersected extent a list is built that contains an indication of each facet of the surface that intersects the extent. A fast mechanism for retrieving the list corresponding to a given extent is therefore necessary. A hash-table is a suitable data structure since it provides constant time access to its elements provided a good hash-code function can be devised which is the case in this application. Indeed the problem has spatial coherency: the surface intersects a limited connected subset of the extents contained in the volume. Therefore a modular mapping of the three coordinates of the extents provides a very good hash function.

For instance, let us consider a 3D image volume divided in cubic 32-pixel-sided extents. Let us assume also that most extracted surfaces are smaller than 1024x1024 pixels. That means most

1. Once the cache is full, each newly read extent replaces the least recently used one in the cache

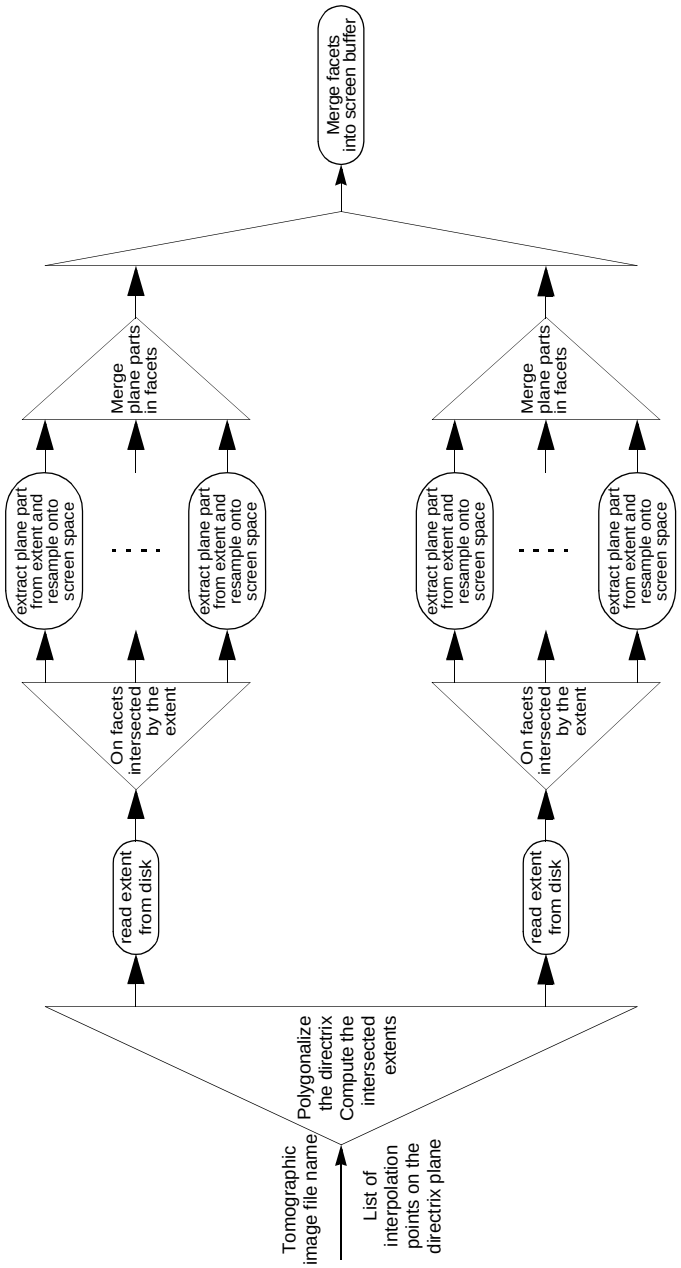


Figure 8-13: Alternative pipelined/parallel surface extraction flow-chart

extracted surfaces are comprised in a 32-extent-sided cube. Therefore we allocate a hash-table with 32^3 entries (128 KB on a system with 32 bits pointers). The extent of coordinates (x, y, z) is stored at index $32^2(z \bmod 32) + 32(y \bmod 32) + (z \bmod 32)$. This index can be computed efficiently with bit-level boolean operations. All images smaller than 1024x1024 pixels are guaranteed to store at most one extent per hash-table entry, while images of sixe 2048x2048 or smaller may store at most two extents per hash-table entry and so on. This shows that even for

```
int ComputeSurfaceExtents(SurfaceParameters* parameters,
                          ExtentReadRequest* request)
{ // C++ code }

void EmptyMerge(void*, SurfaceID*)
{ // Placeholder merge function }

int SelectNextExtentFacet(ReadExtent* extent, ExtentFacetExtraction* request)
{ // For each read extent, generate the necessary plane extraction requests }

void MergeFacetParts(SurfaceFacets* facets, FacetPart* facetPart)
{ // Merge a plane part into the appropriate facet buffer }

leaf operation FacetPartExtraction
   in ExtentFacetExtraction* request
   out FacetPart* facetPart
{ // C++ code }

leaf operation CombineFacets
   in SurfaceID* input
   out SurfaceView* output)
{ // C++ code }

operationPs2Server::SurfaceExtraction
   in    SurfaceParameters* parameters
   out   SurfaceView* surface
{
   parallel while (ComputeSurfaceExtents,
                   EmptyMerge, Client, SurfaceID* surface)
   (
      ExtentServer[thisTokenP->ExtentServerIndex].ReadExtent
      >->
      parallel while (SelectNextExtentFacet,
                      MergeFacetPart, Client, SurfaceFacets facets)
      (
         ComputeServer[thisTokenP->ExtentServerIndex*
                       NB_OF_NODES/NB_OF_DISKS].FacetPartExtraction
      )
   )
   >-> CombineFacets;
}
```

Figure 8-14: CAP pseudo-code for the alternative surface extraction strategy

large extracted surfaces, this hash-table solution should provide better access times than any other data structure.

8.4. Summary

This section presented a novel method for extracting ruled surfaces (more specifically generalized cylinders) from 3D voxel-based volumes such as those produced by most medical imaging modalities. The extraction of cylinders that can be “flattened” and displayed on a 2D display while still preserving lengths, considerably enhances the possibilities offered by oblique plane slicing which remains the fundamental tool for medical image visualization. This extension is intuitive and simplifies user interaction.

The algorithm is based on a definition of digital cylinders that best fit a continuous cylinder by means of adjacent digital plane patches. Digital cylinders can thus be extracted using a digital plane scanning algorithm which was shown to be particularly efficient in the previous chapter. The algorithm can also be parallelized in order to be able to work with very large 3D volumes. Two parallelization strategies were presented: a first one that favors code reuse by relying on the parallel digital plane scanning algorithm and a second one that optimizes the access to the volumic extents making up the 3D volume.

9 Conclusion

Discrete geometry is a new theoretical framework dealing with geometric objects consisting of denumerable sets of points such as those that are generated and manipulated by computers. This discipline is at crossroads between pure abstract mathematics and concrete computer based applications for which it brings an interesting alternative to algorithms based on euclidean geometry. The present research work, divided into two parts, illustrates this situation.

First, we presented a collection of theoretical results related to current problems in discrete geometry. Discrete geometry and more particularly, the subfield called *arithmetic geometry* which we focused on, are indeed new subjects of research where wide areas remain to be explored. Among the contributions, we presented a new approach to the study of 3D digital lines. Studying 3D digital lines is significantly more complex than 2D digital lines. Thanks to this new viewpoint we could derive a definition and interesting theorems about the combinatorial structure of 3D digital lines. Then we introduced a new criterion to polygonalize Bézier curves and surface patches efficiently and in a consistent way with existing results in discrete geometry. Unlike previous approaches this criterion does not rely on an arbitrary precision constant but only on the geometry of the lattice of integers and the definition of naive digital lines and planes. Finally we considered the multi-scale discreteness problem which deals with establishing the relations between the discretizations of geometric objects at different resolutions. Using two different methods, we presented a solution to the determination of the covering of digital lines and parallelograms by regular rectangular tesselations of the plane. While the method used for digital lines was purely arithmetic, the method used for digital parallelograms was geometric, based on the morphological dilation operation, thus building the first bridge between mathematical morphology and discrete geometry.

In the second half of this work, we illustrated the benefits of using discrete geometry for computer imaging with two applications: the extraction of oblique planes and ruled surfaces from 3D discrete images such as tomographic images commonly found in medical imaging. We showed the importance of such applications for radiologists. The extraction of oblique slices is based on a naive digital plane scanning algorithm using integer arithmetic and avoiding costly tri-linear floating point operations. This algorithm was shown to be suitable for parallelization using the CAP/PS^2 framework. The resulting full-software implementation is highly scalable and able to run on architectures ranging from a single isolated PC with one or several disks to a network cluster of PCs with up to 12 disks each. The measured performance figures for the oblique slice extraction showed that the extraction algorithm itself is not the bottleneck in the considered parallel architectures. Furthermore the presence of this application on the Internet in the spectacular Visible Human Slice Server for more than seven months at the time of this writing, has also demonstrated its high stability which can be attributed to some extent to the underlying usage of discrete geometry. The extraction of digital generalized cylinders was pre-

sented as a natural and useful extension of the oblique slicing algorithm. Digital generalized cylinders were defined as the discrete counterpart of euclidean ruled surfaces having a 2D spline for directrix and rulings orthogonal to the plane of the directrix. We showed that these objects could be equivalently considered as an 18-connected juxtaposition of naive digital plane pieces and derived from this result an extraction algorithm based on the naive digital plane scanning algorithm. We also showed that thanks to a careful choice of the geometrical parameters of the plane pieces, additional resamplings can be avoided. Furthermore a parallelization strategy that optimizes the number of accesses to the disks has been proposed. Thus the extraction of digital generalized cylinders incurs a relatively small overhead when compared to the simpler oblique slicing algorithm.

While this work tries to bring solutions to some problems, it also raises various questions and opens the doors for future research. On the theoretical side, none of the subjects that have been considered here has been fully explored. The theory of 3D digital lines is certainly the most ambitious and has room for a lot of new developments: can we define arithmetically the digital line that corresponds to the digitization of an euclidean line by the closest integer point ? How can we control the connectivity of a digital line from its projection on its normal plane ? Can subsets of the projection lattice of $\mathbb{Z}^2$ other than squares define connected subsets of $\mathbb{Z}^3$? Can those subsets be called digital lines ? Can the results be extended to higher dimensions ? The approach to the polygonalization of Bézier surface patches has also left some open questions, especially with respect to the connectivity of adjacent patches. Eric Andrès et al. have provided some directions for 6-connected patches [4] but a more in-depth study is needed for thinner ones. Multi-scale geometry is also an interesting field for further investigations and the link we sketched between mathematical morphology and discrete geometry might be a fruitful research direction.

The applications we developed in this work are a clear example of the benefits of using discrete geometry for digital image processing algorithms. The extraction of more complex surfaces is a natural extension that comes to mind. Depending on how these surfaces are defined, the problem can be more or less intricate. The extraction of digital surfaces relies on a representation of surfaces by digital planar facets, therefore for general digital surfaces, a polyhedrization algorithm is needed. Isabelle Debled has accomplished a reference work in that field [15]. The proposed algorithm remains a little cumbersome however, possibly given appropriate restrictions, a simpler version could be derived. Discrete geometry could find a niche in other fields of computer graphics or medical imaging, e.g., surface tracking, tomographic reconstruction, volume visualization, ray-tracing, etc... In all these fields, discrete approaches have been proposed (cf. the pioneering work of A. Kaufman and his team on discrete ray tracing [57]). These domains could certainly find great benefits in the recent developments of arithmetic geometry as would be the case for other applications involving sampled data, not necessarily images.

Bibliography

1. J. Amanatides, *Ray-tracing with cones,* SIGGRAPH 84, 129-135
2. J. Amanatides and A. Woo, *A Fast Voxel Traversal Algorithm for Ray Tracing,* Proceedings of the 8th Eurographics Conference, Amsterdam, NL, August 24-28, 1987, 3-10
3. T. A. Anderson, C. E. Kim, *Representation of Digital Line Segments and Their Preimages,* Computer Vision, Graphics, and Image Processing, 30, 1985, 279-288
4. E. Andrès, C. Sibata, R. Acharya, *Supercover 3D Polygon,* Discrete Geometry for Computer Imagery, Proceedings of the 6th International Workshop, DGCI'96, Lyon, France, Nov. 1996, Lecture Notes in Computer Science 1176, S. Miguet, A. Montanvert, S. Ubeda Eds., Springer Verlag, 237-242
5. G. Bertrand, R. Malgouyres, *Some Topological Properties of Discrete Surfaces,* Discrete Geometry for Computer Imagery, Proceedings of the 6th International Workshop, DGCI'96, Lyon, France, Nov. 1996, Lecture Notes in Computer Science 1176, S. Miguet, A. Montanvert, S. Ubeda Eds., Springer Verlag, 325-336
6. J.E. Bresenham, *Algorithm for Computer Control of a Digital Plotter,* IBM Systems Journal, 4(1), 1965, 25-30
7. M. P. Do Carmo, *Differential Geometry of Curves and Surfaces,* Prentice-Hall, ISBN 0-13-212589-7
8. S.-L. Chang, M. Schantz and R.Rochetti, *Rendering Cubic Curves and Surfaces with Integer Adaptive Forward Differencing,* Computer Graphics, Vol. 23, Nr 3, Jul. 1989, 157-166
9. J.-L. Coatrieux, C. Barillot, *A Survey of 3D Display Techniques to Render Medical Data,* NATO ASI Series, Vol. F 60, 3D Imaging in Medicine, K. H. Hohne et al Eds.., Springer-Verlag, 1990, 175-195
10. Z.-H. Cho, J. E. Jones, M. Singh, *Foundations of Medical Imaging,* John Wiley and Sons, ISBN 0-471-54573-2
11. D. Cohen-Or, A. Kaufman, *Fundamentals of Surface Voxelization,* Graphical Models and Image Processing, Vol. 57, No. 6, November 1995, 453-461
12. D. Cohen-Or, A. Kaufman, *Scan-Conversion Algorithms for Linear and Quadratic Objects,* Volume Visualization, A. Kaufman Ed., IEEE Soc. Press, 1991, 280-301
13. D. Cohen-Or, A. Kaufman, *3D Line Voxelization and Connectivity Control,* IEEE Computer Graphics & Applications, Vol 17, No 6, Nov-Dec 1997, 80-87

14. M. Coster, J. L. Chermant, *Précis d'analyse d'images*, Presses du CNRS, ISBN 2-87682-020-X

15. I. Debled-Rennesson, *Etude et reconnaissance des droites et plans discrets*, Thèse de Doctorat no 2234, Université Louis Pasteur, Strasbourg, France, Dec. 1995

16. I. Debled-Rennesson, J.-P. Reveillès, *A new approach to digital planes,* SPIE Vol. 2356, Vision Geometry III, 1994, 12-21

17. L. Dorst, A. W. M. Smeulders, *Discrete Representation of Straight Lines,* IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No. 4, July 1984, 450-463

18. T. Elvins, *A Survey of Algorithms for Volume Visualization,* Computer Graphics, Vol. 26, No. 3, Aug. 1992, 40-47

19. G. Farin, *Curves and Surfaces for Computer Aided Geometric Design, A Practical Guide*, Academic Press

20. J. Françon, *Discrete Combinatorial Surfaces,* Graphical Models and Image Processing, Vol. 57, No. 1, Jan. 1995, 20-26

21. L. H. de Figueiredo, *Adaptive Sampling of Parametric Curves,* Graphics Gems V, Academic Press

22. O. Figueiredo, J.-P. Reveillès, *New Results about 3D Digital Lines,* Vision Geometry V, Robert A. Melter, Angela Y. Wu, Longin Latecki, Editors, Proc. SPIE 2826, Aug 96, Denver CO, 98-108

23. J. D. Foley, A. van Dam, S. K. Feiner and J. F. Hughes, *Computer Graphics : Principles and Pratice,* Addison-Wesley Publishing Company

24. J. Françon, *On Recent Trends in Discrete Geometry in Computer Science,* Discrete Geometry for Computer Imagery, Proceedings of the 6th International Workshop, DGCI'96, Lyon, France, Nov. 1996, Lecture Notes in Computer Science 1176, S. Miguet, A. Montanvert, S. Ubeda Eds., Springer Verlag, 3-16

25. R. Fraser, *Interactive Volume Rendering using Advanced Graphics Architectures,* Silicon Graphics Computer Systems, Advanced Systems Division, Mountain View, Ca.

26. B. A. Gennart, B. Krummenacher, L. Landron, R. D. Hersch, *GigaView Parallel Image Server Performance Analysis",* Proceedings of the World Transputer Congress, DeGloria et al. eds., Sep. 1994, 120-135

27. B. A. Gennart, J. Tarraga, R. D. Hersch, *Computer-assisted generation of PVM/C++ programs using CAP,* Proc. Parallel Virtual Machine, EuroPVM'96, LCS 1156, Springer Verlag, 259-269

28. B. A. Gennart, M. Mazzariol, V. Messerli, R. D. Hersch, *Synthesizing parallel imaging applications using the CAP Computer-Aided-Parallelization tool,* IS&T/SPIE's 10th Annual Symposium, Electronic Imaging'98, Storage & Retrieval for Image and Video Databases VI, Jan. 1998, 446-458

29. G. H. Hardy, E. M. Wright, *An Introduction to the Theory of Numbers, 5th Edition,* Osxford Science Publications, ISBN 0 19 853171 0

30. G. T. Herman, *Discrete Multidimensional Jordan Surfaces,* CVGIP, Graphical Models and Image Processing, Vol. 54, No. 6, Nov. 1992, 507-515

31. R. D. Hersch, *Parallel Storage and Retrieval of Pixmap Images,* Proc. of the 12th IEEE Symposium on Mass Storage Systems, IEEE Press, 1993, 221-226

32. A. Kaufman, E. Shimony, *3D Scan-Conversion Algorithms for Voxel-Based Graphics,* Proceedings of the 1986 Workshop on Interactive 3D Graphics, Frank Crow and Stephen M. Pizer Eds, ACM Siggraph, Chapel Hill, NC, October 23-24 1986, 45-75

33. C. E. Kim, *Three-Dimensional Digital Line Segments,* IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 2, March 1983, 231-234

34. R. V. Klassen, *Intersecting Parametric Cubic Curves by Midpoint Subdivision,* Graphics Gems IV, 261-277, Academic Press, 1994

35. T.Y. Kong, A. Rosenfeld, *Digital Topology, Introduction and Survey,* Computer Vision, Graphics and Image Processing, 48, 1989, 357-389

36. V. A. Kovalesky, *New Definition and Fast Recognition of Digital Straight Arcs,* Proc. of the 10th International Conference on Pattern Recognition, Atlantic City, NJ, June 16-21, 1990

37. D. M. Kramer, L. Kaufman, R. J. Guzman, C Hawryszko, *A General Algorithm for Oblique Image Reconstruction,* IEEE Computer Graphics & Applications, No 10, March 1990, 62-65

38. S.Lang, *Algebra,* Addison-Wesley, 1994, 3rd edition

39. J.M. Lane and R. F. Riesenfeld, *A theoretical development for the computer generation and display of picewise polynomial surfaces,* IEEE Trans. Pattern Anal. Machine Intelligence, 2(1), 35-46, 1980

40. M. Levoy, *Display of Surfaces from Volume Data,* IEEE Computer Graphics & Applications, May 1988, 29-37

41. R. Machiraju, R. Yagel, *Accuracy Control of Reconstruction Errors in Volume Slicing,* Biomedical Visulization'95, M. Loew and W. Gershon Ed., Atlanta, GA, Oct. 1995, 50-57

42. R. Malgouyres, *Une définition de surfaces de Z^3,* Ph.D. Thesis, University of Clermont-Ferrand, France, Feb. 1994

43. V. Messerli, B. A. Gennart, R. D. Hersch, *Performances of the PS^2 Parallel Storage and Processing System for Tomographic Image Visualization,* Proceedings of the 1997 International Conference on Parallel and Distributed Systems, Seoul, Korea, December, IEEE Computer Society Press, 514-522

44. V. Messerli, *Tools for parallel I/O and compute-intensive applications,* Ph.D. Thesis Nr. 1915, Ecole Polytechnique Fédérale de Lausanne, Switzerland, Nov. 1998

45. C. E. Mosher, T. Van Hook, *A Geometric Approach for Rendering Volumetric Data,* NCGA'90, Vol. 1,184-193

46. J.-P. Reveillès, *Géométrie discrète, calcul en nombres entiers et algorithmique,* Thèse d'Etat, Université Louis Pasteur, Strasbourg, Dec 1991

47. M. L. Rhodes, W. V. Glenn, Y. M. Azzawi, *Extracting oblique planes from serial CT sections,* Journal of Computer Assisted Tomography, Vol. 4, 1980, 649-657

48. A. Rosenfeld, *Digital Straight Line Segments,* IEEE Transactions on Computers, Vol. c-23, No. 12, Dec. 1974, 1264-1269

49. D. Ruprecht and H. Müller, *Deformed Cross-dissolves for Image Interpolation in Scientific Visualization,* The Journal of Visualization and Computer Animation, Vol. 5, 1994, 167-181

50. J. Serra, *Introduction to Mathematical Morphology,* Computer Vision, Graphics and Image Processing, 35, 1986, 283-305

51. J. P. Singh, A. Gupta, M. Levoy, *Parallel Visualization Algorithms: Performance and Architectural Implications,* IEEE Computer, Vol. 27, No. 7, Jul. 1994, 45-55

52. M. R. Stytz, G. Frieder, O. Frieder, *Three-Dimensional Medical Imaging: Algorithms and Computer Systems,* ACM Computing Surveys, Vol. 23, No. 4, Dec. 1991, 421-499

53. J. K. Udupa, *3D Discrete Space,* Volume Visualization, A. Kaufman Ed., IEEE Soc. Press, 1991, 241-243

54. S. Vetsch, V. Messerli, O. Figueiredo, B. Gennart, R. D. Hersch, L. Bovisi, R. Welz, L. Bidaut, *A Parallel PC-based Visible Human Slice Web Server,* Proceedings of the Visible Human Project Conference, 1998, Oct. 1-2, Bethesda, Maryland USA, sponsored by the National Library of Medicine. Available on CD-ROM. `http://www.nlm.nih.gov/research/visible/visible_human.html`

55. B. Wallis, *Tutorial on Forward Differencing,* Graphics Gems, Andrew S. Glassner Ed., Academic Press

56. G. Wang, W. Xu, *The Termination Criterion for Subdivision of the Rational Bézier Curves,* CVGIP: Graphical Models and Image Processing, Vol. 53, No. 1, January, pp. 93-96, 1991

57. R. Yagel, D. Cohen-Or and A. Kaufman, *Discrete Ray Tracing,* IEEE Computer Graphics and Applications, Vol 12, No 5, Sept. 1992, 19-28

Biography

Oscar Figueiredo was born on July 21st, 1971 in Murtede, Portugal. He is now a portuguese and french citizen living in Lausanne, Switzerland.

He graduated top of his year as a Physics and Electronics Engineer with a specialization in Computer Science at CPE Lyon (formerly ICPI) in 1994. He also holds a Master Degree (D.E.A.) in Computer Graphics from the University and Ecole des Mines of St Etienne (France) he received on the same year. He has been working at the Peripheral Systems Laboratory of the Ecole Polytechnique Fédérale de Lausanne on discrete geometry and its applications in the field of medical imaging during the past four years.

His past professional experience includes:

- the development of a multi-platform software framework for surface texture analysis at Digital Surf in Besançon, France (`http://www.digitalsurf.fr/`) in 1994
- the development of a 32 kHz benchmark system for deviation coils for high-definition CRTs at Thomson Tubes & Displays, Genlis, France in 1993
- various jobs as a worker, including 2 months in a restaurant in New Jersey, USA in 1992

Publications

- Oscar Figueiredo, Jean-Pierre Reveillès and Roger D. Hersch
  Digitization of Bezier Curves and Patches using Discrete Geometry
  DGCI'99, March 1999, to be published in Springer Verlag's LNCS Series
- S. Vetsch, V. Messerli, O. Figueiredo, B. Gennart, R. D. Hersch, L. Bovisi, R. Welz, L. Bidaut
  A Parallel PC-based Visible Human Slice Web Server
  Proceedings of the Visible Human Project Conference, 1998, Oct. 1-2, Bethesda, Maryland USA, sponsored by the National Library of Medicine. Available on CD-ROM.
  `http://www.nlm.nih.gov/research/visible/visible_human.html`
- Oscar Figueiredo, Jean-Pierre Reveillès
  New Results about 3D Digital Lines
  Vision Geometry V, Robert A. Melter, Angela Y. Wu, Longin Latecki Editors, Proc. SPIE 2826, 08/96, Denver CO, pp 98-108
  `http://diwww.epfl.ch/w3lsp/pub/publications/other/`

- O. Figueiredo, J.P. Reveillès
  A Contribution to 3D Digital Lines
  Proceedings of the 5th Colloquium on Discrete Geometry and Computer Imagery, DGCI'96, September 25-27, 1995, Clermont-Ferrand, pp. 187-198
  `http://diwww.epfl.ch/w3lsp/pub/publications/other/`

Other Reference

- The Visible Human Slice Server, `http://visiblehuman.epfl.ch/`