Ann. Univ. Ferrara - Sez. VII - Sc. Mat.
Supplemento al Vol. XLV, 1-0 (2000)

xcmodel: an aCADemic system (*)

G. CASCIOLA(**) - S. MORIGI(***)

SUNTO - *xcmodel* è un sistema CAD realizzato e utilizzabile in ambito accademico. E' composto da quattro pacchetti: un modellatore 2D, un modellatore 3D, un compositore di oggetti e un pacchetto per la resa realistica di scene; questi sottosistemi sono in costante evoluzione. Il sistema riassume le nostre competenze ed esperienze nella modellazione geometrica e sulle curve e superfici NURBS acquisite in oltre dieci anni di ricerca. *xcmodel* e i suoi sottosistemi sono stati progettati per essere un laboratorio di ricerca e di didattica, per sperimentare e imparare; è un ambiente ideale per sviluppare, mettere a punto e confrontare metodi ed algoritmi della modellazione geometrica e della grafica.

ABSTRACT - *xcmodel* is a CAD system realized and usable in an academic environment. It integrates four packages: a 2D and a 3D modeller, an object composer and a realistic scene renderer; these subsystems can be regarded as being in constant evolution i.e. a continuous work in progress. The system summarises our knowledge and experience in geometric modelling and NURBS curves and surfaces acquired over ten years of research. *xcmodel* and its subsystems were designed to represent a research and teaching laboratory to experiment and learn; it is an ideal environment to develop, perfect and compare methods and algorithms in geometric modelling and graphic visualization.

1. – Introduction

xcmodel is an interactive graphics system based on NURBS (Non Uniform Rational B-Splines) with the power of a professional CAD (Computer Aided Design) system, but realized and usable in an academic environment.

xcmodel was designed to experiment and evaluate new and well-known methods for modelling and graphics visualization. The system has been developed and

(*)This work was supported by MURST, Cofin97, Numerical Analysis: Methods and Mathematical Software.
(**)Indirizzo dell'autore: Department of Mathematics, University of Bologna, P.zza di Porta S.Donato 5, Bologna, Italy. casciola@dm.unibo.it
(***)Indirizzo dell'autore: Department of Mathematics, University of Bologna, P.zza di Porta S.Donato 5, Bologna, Italy. morigi@dm.unibo.it.

grown with the contribution of many graduates and undergraduates in mathematics and computer science. Thus the system has been adopted in many university courses in geometric modelling and computer graphics as a laboratory to experiment and learn. One of the educational peculiarities of *xcmodel* making it different from a commercial system which uses a high level design philosophy, is that it assumes the users are familiar with the mathematics of the representations used in the modelling system and let them interact with every single operator or parameter governing the shape of the object he/she has in mind. As a consequence, we expect the user to be a researcher or a student. *xcmodel* integrates four packages: a 2D and a 3D free form modeller, an object composer and a realistic scene renderer. They cover the main steps in the modelling and rendering process. These subsystems can be regarded as being in constant evolution, i.e. a continuous work in progress.

The following sections do not intend to be exhaustive in this description of the wide variety of operators and algorithms for NURBS geometric modelling and rendering. They focus on those new and well-known methods that are implemented and experimented in *xcmodel*. Moreover, we will not go into detail about the mathematics or algorithms; rather the reader will be referred to an extensive bibliography for more detailed information.

Section 2 describes the requirements of the *xcmodel* system. In Section 3 some basic definitions about the modelling primitives involved in the system are given; Section 4 considers some geometric tools that are fundamental for the more complex modelling techniques described in Section 5. Section 6 deals with the description of a virtual scene. Finally the rendering methods implemented in *xcmodel* are presented in Section 7.

2. – *xcmodel* requirements

The *xcmodel* functional structure, in Figure 1, shows how the four main interactive graphic subsystems are integrated:

- *xccurv*: performs the modelling of 2D NURBS curves [21];
- *xcsurf*: performs the modelling of 3D NURBS curves and surfaces [22];
- *xcbool*: performs boolean operations on solids [23];
- *xcrayt*: performs the description and rendering of modelled scenes by a ray-tracing algorithm [24].

The ellipses represent *xcmodel* input/output data. The arrows indicate the data-flow, while the rectangular blocks represent the main subsystems which are totally independent of one another. The dashed arc separates the modelling part (left) from the rendering part (right).

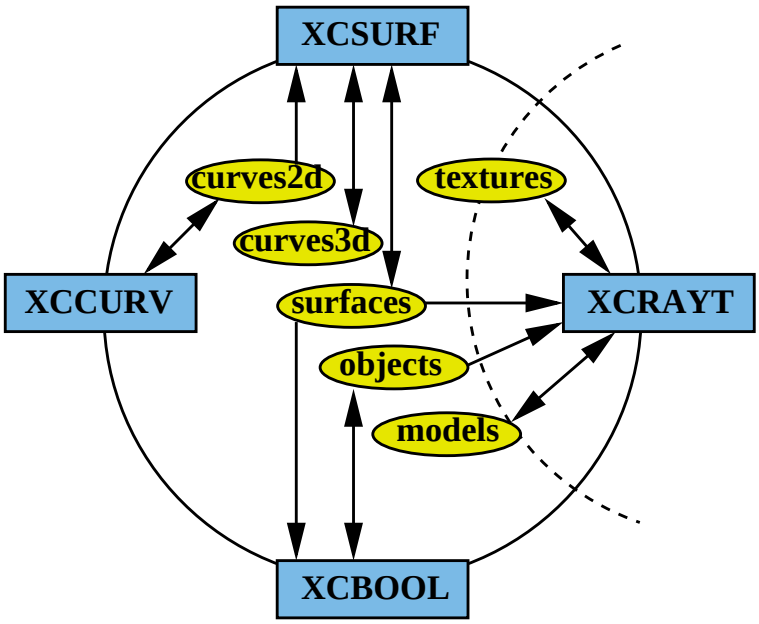

Figure 1: *xcmodel* functional scheme

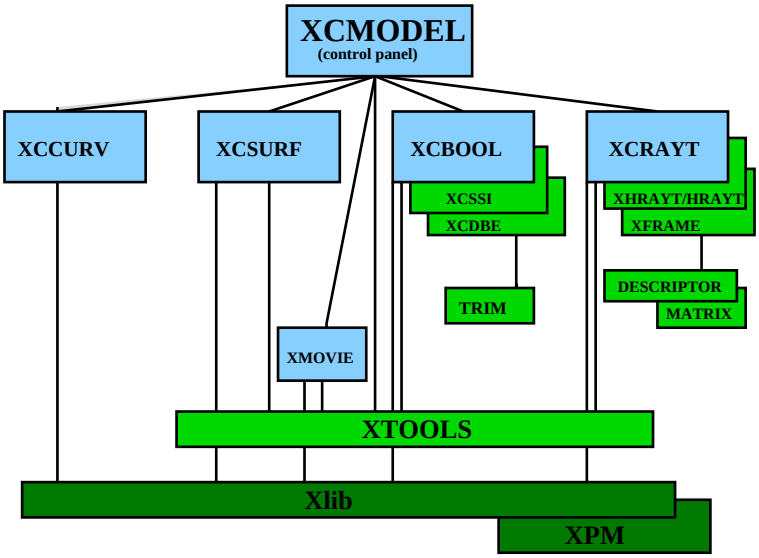

Figure 2: *xcmodel* system architecture

At the present time, the *xcmodel* system runs on UNIX platforms such as Sun SPARC (Solaris), SGI (Irix), and Intel (Linux), but we expect problems to be minimal under other environments. The system is a collection of modules and libraries entirely implemented in C ANSI programming language.

Figure 2 shows the *xcmodel* system architecture. The graphical user interface of the environment is realized using the *xtools* library [28], built on the top of Xlib, the lowest level library of the XWindow System. The graphic icons in the user interface make use of the XPM (XPixMap) library which allows us to deal with pixmaps. Together with the *xtools* library, other three libraries have been implemented in the system: the *MATRIX* library [25], for vector and matrix computation, the *descriptor* library [26], for scene setting, and the *trim* library [27] for the visualization of trimmed NURBS surfaces in real time. The scene descriptor and rendering *xcrayt* package makes use of two modules: *xhrayt/hrayt* for ray-tracing and *xframe* for visualizing the rendered scene. The solid composer *xcbool* module makes use of two independent subsystems: *xcssi*, for surface/surface intersection and *xcdbe*, a conversion tool for trimmed surfaces. Animation is managed by *xmovie* tool [24] that allows for the visualization of images and image sequences. The *xcmodel* software package is available [20] and a freely distributed version (*xcmodel* ver. 1.0) as well as user's guides, data, models and images, can be downloaded from the web page:

http://www.dm.unibo.it/∼casciola/html/xcmodel.html

3. – *xcmodel* modelling primitives

This section provides some basic definitions of the NURBS modelling primitive on which *xcmodel* is based. The system assumes NURBS as the unique canonical form for modelling.

To define rational B-splines we make use of homogeneous coordinates. If $P = (x, y, z)$ is a point in 3D Euclidean space, we denote a corresponding point in 4D homogeneous space by $P^w = (wx, wy, wz, w)$, where $w > 0$. A B-spline curve of order p in homogeneous space can be defined by the equation

$$\mathbf{c}^w(t) = \sum_{i=1}^{p+L} \mathbf{P}_i^w N_{i,p}(t)$$

where $\mathbf{P}_i^w$ are the control points in homogeneous space,

$$T = (\underbrace{t_1, \cdots, t_p}_{p}, t_{p+1}, \cdots, t_{p+L}, \underbrace{t_{p+L+1}, \cdots, t_{2p+L}}_{p})$$

is a nondecreasing sequence of real numbers called knots, and $N_{i,p}(t)$ are the order p B-splines defined over the knot vector T. The NURBS curve $\mathbf{c}(t)$ associated with $\mathbf{c}^w(t)$ is obtained by the projection of $\mathbf{c}^w(t)$ into a 3D space, that is, by dividing the first three coordinates of each point by its homogeneous coordinate, thus obtaining the following definition:

$$\mathbf{c}(t) = \frac{\sum_{i=1}^{p+L} w_i \mathbf{P}_i N_{i,p}(t)}{\sum_{j=1}^{p+L} w_j N_{j,p}(t)} = \sum_{i=1}^{p+L} \mathbf{P}_i R_{i,p}(t),$$

with

$$R_{i,p}(t) = \frac{w_i N_{i,p}(t)}{\sum_{j=1}^{p+L} w_j N_{j,p}(t)}$$

rational basis functions. Projections of NURBS surfaces of order (m, n) are obtained analogously

$$\mathbf{r}^w(u,v) = \sum_{i=1}^{n+H} \sum_{j=1}^{m+K} \mathbf{P}_{ij}^w N_{i,n}(u) N_{j,m}(v)$$

or, equivalently,

$$\mathbf{r}(u,v) = \frac{\sum_{i=1}^{n+H} \sum_{j=1}^{m+K} w_{ij} \mathbf{P}_{ij} N_{i,n}(u) N_{j,m}(v)}{\sum_{i=1}^{n+H} \sum_{j=1}^{m+K} w_{ij} N_{i,n}(u) N_{j,m}(v)} = \sum_{i=1}^{n+H} \sum_{j=1}^{m+K} \mathbf{P}_{ij} R_{i,n;j,m}(u,v)$$

$$R_{i,n;j,m}(u,v) = \frac{w_{ij} N_{i,n}(u) N_{j,m}(v)}{\sum_{i=1}^{n+H} \sum_{j=1}^{m+K} w_{ij} N_{i,n}(u) N_{j,m}(v)}$$

where $w_{ij} > 0$ are the weights, $\mathbf{P}_{ij}$ are the control points,

$$U = (\underbrace{u_1, \cdots, u_n}_{n}, u_{n+1}, \cdots, u_{n+H}, \underbrace{u_{n+H+1}, \cdots, u_{2n+H}}_{n}),$$

$$V = (\underbrace{v_1, \cdots, v_m}_{m}, v_{m+1}, \cdots, v_{m+K}, \underbrace{v_{m+K+1}, \cdots, v_{2m+K}}_{m}).$$

are nondecreasing sequences of knots, and $N_{i,n}(u), N_{j,m}(v)$ are the order n and m B-splines, respectively, defined over the knot vectors U and V.

For a detailed discussion of the properties of NURBS curves and surfaces we refer the reader to [66].

One of the main reasons for the choice of NURBS for *xcmodel* is their wide range of expression. A detail discussion of the widespread acceptance and popularity of NURBS in CAD/CAM and graphics systems is given in [62].

xcmodel has general characteristics, which is unusual for commercial CAD systems. For example, *xcmodel* allows the user to set the curve (surface) order

up to a value of ten. As a result the algorithms implemented are as general as possible. *xcmodel* provides some predefined knot vectors such as "equally spaced", "uniform" (the interior knots are equally spaced and the exteriors coincide), "periodic", "chord length" [35], but also "manually", where the user sets any knot position and multiplicity. *xcmodel* manages collapsed control points and the relative degenerate parametrizations. The rational curves and surfaces are computed via non-rational curves and then by projection; a spline is computed by its B-splines representation, applying the well known recurrence relation provided by Cox-de Boor [11, 33].

The most useful NURBS paradigm is limited by the requirement that the surfaces are defined over rectangular regions and this leads to topological rectangular patches. This limit of NURBS surfaces is overcome by the trimmed NURBS, that is NURBS surfaces defined on arbitrary restricted parametric domains. A trimmed NURBS surface can be defined by a NURBS surface and a set of trimming curves in the parametric space of the surface. This set of planar, closed, non-intersecting curves can be conveniently represented as NURBS curves, which are simply reduced to a closed polygonal when the degree is equal to 1.

A trimmed NURBS surface is given by the restriction of $\mathbf{r}(u, v)$ to a subdomain $D \subset U \times V$ of the parametric space, named *trimmed region*. This domain D is defined as the set of regions on $U \times V$ whose boundaries are specified by trimming curves. This allows us to identify the part of the surface that remains when discarding all the holes defined by the trimming curves.

In *xcmodel* the trimmed domain is represented by a 2D CSG tree equivalent to the CSG tree presented in [36, 13], but differently represented. Each node of the CSG tree structure is an embedded non-intersecting closed curve that defines a limited region built starting from the 2D tree. Regions at the same level are disjoint to each other. At alternate levels these regions are combined using union and difference boolean operators. The trimmed region looks like a set of islands and lakes, where the islands represent part of the trimmed region, while the lakes are the holes in it. The algorithm used, forms a union of islands (R_i), and subtracts from each island all of its lakes (S_{ij}) starting from a 2D tree:

$$D = \bigcup_{i=1}^{I} (R_i \bigcap (\bigcup_{j=1}^{L_i} S_{ij})),$$

where I is the number of islands and L_i is the number of the lakes S_{ij} children of island i. The trimmed region that is obtained can be determined by classifying a single point. Conventions can be assumed to consider alternate levels of the tree to be island and lake regions, starting from an island/lake top level.

Another approach implemented in *xcmodel* for representing a trimmed domain D is based on the idea of decomposing $U \times V - D$ using a list of planar subregions, whose union defines the entire complementary domain $U \times V - D$,

thus allowing the user to manage a set of surfaces, defined on an irregular domain, that does not contain any trimming curves. This approach is implemented as a particular case of a 2D CSG with all first-level children starting from an island top level.

The representation scheme used in *xcmodel* for modelling a solid object is the so-called B-rep (Boundary representation). In this scheme a 'primitive solid' is determined by its boundary, that consists of a single closed surface, in order to separate the space into two parts, one of which will be enclosed. According to convention, the boundary surface must be parametrized in such a way that the surface normal vector indicates the area outside the solid. For example, spheres, cylinders, etc. are included in this definition, as well as closed sculptured solids. Primitive solids can also be combined by boolean operations, such as intersection, union and difference, in order to create a complex solid.

The boundary of a solid object S is defined in terms of trimmed surfaces as follows:

$$bS = \bigcup_{i=1}^{k} \mathbf{r}_i(D_i)$$

where D_i is the trimmed region associated with the surface $\mathbf{r}_i$. A primitive solid is then defined by $k = 1$ and $D_1 = U \times V$. The boundary of the solid S is a closed surface.

As academic system *xcmodel* does not provide any control on the topology of the built objects, thus it does not ensure that the solid created corresponds to a phisically realizable object.

4. – Basic geometric tools

In this section we present four tools for curves which are fundamental in spline curve and surface modelling in general and in *xcmodel* in particular; these are knot-insertion, knot-removal, degree elevation and reparametrization. Note that all these tools only affect curve or surface representations, while geometrically the curves or surfaces remain unchanged.

4.1. – *Knot-insertion.* – This is the process of expressing a particular curve $\mathbf{c}^w(t)$ in terms of a related spline representation that has one or more additional knots. This process is also known as refinement, since new knots ($s \geq 1$) are inserted into the original knot vector T to produce a proper refinement $\overline{T}$.
Then control points $\mathbf{Q}_i^w$ exist such that

$$\mathbf{c}^w(t) = \sum_{i=1}^{p+L+s} \mathbf{Q}_i^w \overline{N}_{i,p}(t).$$

In 1980, Boehm [8] and Cohen et al. [29] independently found two different algorithms to compute the new control points $\mathbf{Q}_i^w$. Efficient algorithms for implementing both the special case of single knot insertion [8] and the general case [10, 54] have been proposed. Performances are analyzed in [9, 31]. In *xcmodel* this tool is used in:

- evaluating points and derivatives on curves and surfaces;
- subdividing curves and surfaces (splitting);
- adding control points in order to increase flexibility in shape control (interactive design and hierarchical design);
- decomposition of spline curves and surfaces into their constituent (Bezier) polynomial pieces;
- converting a periodic knot vector into a uniform one;
- merging two or more knot vectors in order to obtain a set of curves which are defined on a common knot vector (see section 5);
- obtaining polygon (polyhedral) approximations to curves (surfaces) by refining knot vectors; this makes the control polygon (net) converge to the curve (surface).

4.2. – *Knot-removal.* – This is the reverse process of knot insertion. We can say that, given a knot vector T, a knot $\overline{t}$ is removable if $\mathbf{c}^w(t)$ has a precise representation in the B-spline basis $\overline{N}_{i,p}(t)$ defined over the knot vector T without $\overline{t}$. A knot-removal algorithm must determine if a knot is removable and how many times ($s \geq 1$), then compute the new control points $\mathbf{Q}_i^w$ such that the new curve representation is

$$\mathbf{c}^w(t) = \sum_{i=1}^{p+L-s} \mathbf{Q}_i^w \overline{N}_{i,p}(t).$$

Details can be found in [75]. In *xcmodel* knot-removal is used to obtain the most compact representation of the curve/surface in:

- adjusting knot vector after control point insertion/displacement;
- joining spline curves together to form composite curves.

4.3. – *Degree elevation.* – This tool allows us to represent a spline curve $\mathbf{c}^w(t)$ as a curve with an elevated degree $s > p - 1$:

$$\mathbf{c}^w(t) = \sum_{i=1}^{s+1+\overline{L}} \mathbf{Q}_i^w N_{i,s+1}(t).$$

Efficient but mathematically complicated methods to compute the new control points $\mathbf{Q}_i^w$ are given by Prautzsch in [68], by Cohen et al. in [30, 32] and by Prautzsch and Piper [69]. The algorithm provided by Prautzsch and Piper is the most efficient for the general case, but Cohen et al. also give simple and efficient algorithms for low-degrees, such as linear to quadratic and quadratic to cubic. All these algorithms raise the degree by 1. In 1994 Piegl and Tiller [64] presented another algorithm which is mathematically simpler, and competitive with that given in [69], particularly in the case where the degree is to be raised by more than 1. In *xcmodel* degree elevation is used to:

- make compatible section curves in the "surface from curves" modelling technique (see section 5.3);
- obtain "bi-p-ic" parametric surfaces (where e.g. p=4 for bicubic) in surfaces from curves modelling techniques (see section 5.3), where the order of the profile and trajectory curves may be required to be the same.

4.4. – *Reparametrization.* – Let $\mathbf{c}(t)$ be a parametric curve on $t \in [a, b]$ and assume that $t = f(s)$ is a scalar-valued function on $s \in [c, d]$ satisfying:

- $f\prime(s) > 0$ for all $s \in [c, d]$,
- $a = f(c)$ and $b = f(d)$,

the composition of $\mathbf{c}(t)$ and $f(s)$, given by $\mathbf{c}(s) = \mathbf{c}(f(s))$ is called a reparametrization of $\mathbf{c}(t)$. $\mathbf{c}(s)$ is geometrically the same curve as $\mathbf{c}(t)$, but parametrically they are different.
In *xcmodel* we consider only reparametrization functions that keep the curve a NURBS. In particular, for a rational linear reparametrization function in [2] and [52], an explicit expression for the reparametrized NURBS curve is given.
The most common and useful parametrization is the arc length. Such a parametrization is unfortunately almost never possible for NURBS, unless for straight lines [37]. Therefore *xcmodel* offers several reparametrization techniques that best fit the arc length parametrization [18].

A reparametrization can be required because of the effects that the change of weights and control points have on the parametrization of the curve. Moreover, any numerical method, or simply the rendering procedure, is affected by their particular parametrization in terms of computational complexity and numerical stability.

5. – Modelling

xcmodel offers a set of design tools for modelling curves, surfaces and solid objects that can be classified into four classes: basic constructors, shape modifiers, surfaces from curves and object composer. Basic constructors build models interactively or automatically, together with the possibility of using, as initial shapes, classical primitive objects, such as spheres, torus, boxes or more complex objects like revolution surfaces. Interactive modelling tools for shape modification aim to modify the shape of an existing object. In addition, we can define operators on curves to create surfaces. Finally, boolean operators are available for building complex objects.

Each of these tools can be used alone or in pipeline fashion in order to get the final geometric model ready to be rendered.

5.1. – **Basic constructors**. – In this section we deal with the capabilities of *xcmodel* called "shape approximation" and "data fitting" to create 2D or 3D shapes ab initio.

- Shape approximation -
B-splines provide a geometrically intuitive basis, one in which the shape of the curve or surface has a predictable and comprehensible relationship to the coefficients (i.e. control points). This is a special approximation scheme named shape approximation, which is extremely attractive for geometric design. According to this approximation scheme, an individual control point only has a local effect, and does not affect a design beyond the localized region of influence. In *xcmodel*, design tools are provided to give 2D points for a control polygon (for curves) and 3D points for a control net (for surfaces) interactively and a set of parameters to define the curve or surface shape approximation of the control polygon or net.

- Data fitting -
In this paragraph we deal with fitting, i.e. the construction of NURBS curves and surfaces which fit a rather arbitrary set of geometric data, such as points and derivative vectors. We distinguish two types of fitting, interpolation and approximation by the least square technique. There are several mathematical methods for interpolating or approximating a single-valued function from a given set of values, but their application to curve fitting sometimes results in a curve that is very different from what the designer intended. Input to a fitting problem consists of geometric data, such as points and derivatives. Output is a NURBS curve or surface, i.e. control points, knots and weights. Furthermore, either the order p (or (m, n) for surfaces) must be the input or the algorithm must select an appropriate order. Very little has been published on setting the weights in the fitting process. Most often, all the weights are simply set to 1. Finally, there are many methods for choosing the knots, most of them heuristic.
The fitting algorithms can be global or local. With a global algorithm, a system of equations is set up and resolved. Since the given data consists only of points

and derivatives, and the control points are the only unknowns (order, knots and weights have been preselected or precomputed), the system is linear and hence easy to solve. Local algorithms are more geometric in nature, constructing the curve or surface segment-wise, using only local data at each step. These algorithms are usually computationally less expensive than global methods and can deal with local data anomalies better; however, achieving desired levels of global continuity is not a standard matter and local methods often result in multiple internal knots. In what follows we present the global and local methods for curve and surface interpolation and approximation implemented in *xcmodel*.

- **Global curve interpolation to point and derivative data**
  Let $\mathbf{Q}_i^l, \quad i = 1, \ldots, n$ and $l = 0, \ldots, l_i$, be the interpolation points and derivative vectors; we want to interpolate these data with a p-th order NURBS curve. Note that if $l_i = 0 \quad \forall i$, then we have a Lagrange interpolation, otherwise we have an Hermite interpolation. If we assign a parameter value $\overline{t}_i$, to each $\mathbf{Q}_i^0$, select an appropriate knot vector T and select a positive weight vector W, we can set up the $(p + L) \times (p + L)$ system of linear equations

$$\sum_{i=1}^{p+L} \mathbf{P}_i R_{i,p}^l(\overline{t}_i) = \mathbf{Q}_i^l \quad i = 1, \ldots, n \quad l = 0, \ldots, l_i$$

  with $L = \sum_{i=1}^{n}(l_i + 1) - p$. The control points $\mathbf{P}_i$ are the unknowns. The problem of choosing the $\overline{t}_i$, T and W remains, and their choice affects the shape and parametrization of the curve. *xcmodel* provides four methods of choosing the $\overline{t}_i$: chord length, uniform, centripetal and exponential [51, 55]. The knot vector T is computed in a heuristic way, so that the Schoenberg-Whitney and Karlin-Ziegler conditions are satisfied in order to have a unique solution for the interpolation problem [58]. The positive weight vector W characterizes a rational interpolation from a non-rational by setting manually or equal weights respectively. In the case of Hermite interpolation, derivative vectors have to be specified as input data. For the more common case of first derivative vectors, *xcmodel* provides the Akima technique [1] for an automatic derivative computation. In the cubic case there is the possibility of interpolating the given points with a periodic curve or with a rational C^1 curve. In the latter case the NURBS weights can be used as tension parameters [44].

- **Global surface interpolation to point data**
  Given a set of $(n + H) \times (m + K)$ data points $\mathbf{Q}_{kl} \quad k = 1, \ldots, n + H$ and $l = 1, \ldots, m + K$, we want to construct a non rational (m, n)-order 3D

spline surface interpolating these points, i.e.

$$\sum_{i=1}^{n+H}\sum_{j=1}^{m+K} \mathbf{P}_{ij} N_{i,n}(\overline{u}_k) N_{j,m}(\overline{v}_l) = \mathbf{Q}_{kl}. \tag{1}$$

In *xcmodel* this is achieved by setting $\mathbf{r}^w(\overline{u}_k, \overline{v}_l) = \mathbf{Q}^w_{kl}$, with all weights equal. Again, the first thing to do is to compute reasonable values for the $(\overline{u}_k, \overline{v}_l)$ and the knot vectors U and V. A common way is to use one of the previous methods for curves to compute parameters $\overline{u}^l_1, \ldots, \overline{u}^l_{n+H}$ for each l, and then to obtain each $\overline{u}_k$ by averaging across all $\overline{u}^l_k$ for $l = 1, \ldots, m+K$ [62]. Using the parameter points computed, we get the U and V vectors in exactly the same way as with curves. Clearly the equations (1) represent a linear system for the unknowns $\mathbf{P}_{ij}$. However, since the surface is tensor product, the $\mathbf{P}_{ij}$ can be obtained more simply and efficiently as a sequence of curve interpolations. At the moment, *xcmodel* provides only bicubic spline surface interpolation.

- **Local curve interpolation to point and derivative data**
  Let $\mathbf{Q}^l_i, \quad i = 1, \ldots, n$ and $l = 0, 1$, be the interpolation points and first derivative vectors given. By local curve interpolation we mean a method which constructs n polynomials or rational curve segments $\mathbf{c}_i(t), \quad i = 0, \ldots, n-1$, such that $\mathbf{Q}^0_i$ and $\mathbf{Q}^0_{i+1}$ are the end points of $\mathbf{c}_i(t)$. Neighboring segments are joined with some prescribed level of continuity, and the construction proceeds segment-wise. *xcmodel* provides, at the moment, only a local method, which interpolates point and first derivative data with a rational globally C^1 (cubic over quadratic). The NURBS weights play the role of tension parameters and for $w_i \to \infty \quad \forall i$, the curve converges to the polygonal defined by the interpolation points. It is a NURBS adaptation of an algorithm by Gregory and Sarfraz [44].

- **Weighted and constrained least square curve fitting**
  Let $\mathbf{Q}_i, \quad i = 1, \ldots, n$, be the points to be approximated. If we assign a parameter value $\overline{t}_i$, to each $\mathbf{Q}_i$, choose a curve order p, an appropriate knot vector T, and select a positive NURBS weight vector W, we can set up the $n \times (p + L)$ overdetermined system of linear equations

$$\sum_{i=1}^{p+L} \mathbf{P}_i R_{i,p}(\overline{t}_i) = \mathbf{Q}_i \quad i = 1, \ldots, n.$$

  Most often, all these weights are simply set to 1. Indeed, for approximation, there is little reason to do otherwise.
  Optionally the first and last given points can be constrained (precisely

interpolated) or the curve can be constrained to be closed and periodic. We also allow a positive weight to be assigned to each unconstrained item. Increasing a weight increases the tightness of the approximation to that item, whereas decreasing the weight looses the approximation to that item. Notice that these weights have nothing to do with weights in the NURBS sense. *xcmodel* also provides a method to approximate the weighted and constrained data in the least-squares sense.

5.2. – **Shape modifiers**. – The purpose of some methods implemented in *xcmodel* is to provide tools which allow a user to interactively make local modification to an existing NURBS curve or surface. A NURBS curve or surface is defined by its control points, weights, knots and orders; modifying one or more of these parameters, can affect and change the shape of the curve or surface [66]. Since *xcmodel* is an educational system and assumes the user are experimenting and learning, it allows the user to modify all these parameters interactively. Generally CAD systems on the market do not allow the user to modify such parameters; they apply many predefined shape operators to the curve or surface. Following [60, 61], *xcmodel* implements some of these operators, such as constraining a curve point to pass over a given point by one control point repositioning, by one weight modifications or by two weight modification, warping, flattening, etc. The result of applying a transformation to the whole spline curve or surface, gives rise to a spline defined by the transformed control points together with the original knot vector and curve (surface) order. Formally

$$\mathbf{c}_A^w(t) = \sum_{i=1}^{p+L} A\mathbf{P}_i^w N_{i,p}(t)$$

is the result of applying the transformation matrix A to the curve $\mathbf{c}^w(t)$. In the case where A is an affine transformation, this strategy produces exactly the same curve (surface) which would have been produced if A had been applied to each point on the curve (surface).

In 1988 Forsey and Bartels [39] proposed the Hierarchical Spline Surfaces (HSS) enabling a surface to be refined and detail added, using a hierarchy, whose levels correspond to the different levels of refinement of the surface patches. This proposal enables us to restrict the influence of refinement to the relevant part of the surface. In *xcmodel* we implemented the HSS idea by using the trimmed NURBS surfaces power and precisely defining a "rectangular trimmed NURBS surfaces hierarchy" [19].

5.3. – **Surface from curves**. – As interactive techniques for the sculpturing of surface shapes on a two dimensional display involve obvious difficulties, additional tools are available for the designer to model the desired surface using operators on curves. The most frequently used techniques, such as skinning, sweeping, and swinging, are adopted in *xcmodel*.

NURBS skinning is a special surface interpolation technique by which an ordered set of NURBS curves (called section curves) is interpolated to form a NURBS surface. The section curves have to be made compatible, i.e. same degree and knot vector, by means of the basic geometric tools described in section 4. If this is the case they will also have the same number of control points, and they can be defined as follows:

$$\mathbf{c}_j^w(u) = \sum_{i=1}^{n+H} \mathbf{Q}_{ij}^w N_{i,n}(u), \quad j = 1, \ldots, m+K.$$

For each index i, their control vertices $\mathbf{Q}_{ij}^w, \quad j = 1, \ldots, m+K$, in v direction are interpolated using global curve interpolation technique in homogeneous space (see section 5.1) obtaining the following curves

$$\mathbf{c}_i^w(v) = \sum_{j=1}^{m+K} \mathbf{P}_{ij}^w N_{j,m}(v), \quad i = 1, \ldots, n+H,$$

that pass through $\mathbf{Q}_{ij}^w$ at certain $\overline{u}_i$ values.

The control polygons of the interpolation curves together form a control mesh that defines a skinned surface:

$$\mathbf{r}(u,v)^w = \sum_{i=1}^{n+H} \sum_{j=1}^{m+K} \mathbf{P}_{ij}^w N_{i,n}(u) N_{j,m}(v).$$

The above skinning algorithm is a fairly straightforward generalization to rational curves of the skinning algorithm for polynomial B-spline curves described in [78, 79], which has the desirable property that the created surface is as smooth as the section curves. Now, as for the rational case, the algorithm works in the homogeneous space, even if, for example, a rational curve is C^1 in $3D$, its associated curve in $4D$ may be only C^0, that is, the algorithm can produce a C^0 surface in $4D$ whose associated surface in $3D$ will exhibit discontinuities. This smoothness problem is addressed in [47] and [62]. The positioning of the section curves in the space is another critical problem discussed in [79]: unevenly positioned section curves can lead to unsatisfactory surface shapes.

NURBS sweeping technique is a special case of skinning that uses a constant section curve and sweeps it along another curve (called spine or trajectory curve) in order to create a NURBS surface. The positioning points for sweeping can be chosen as the spine knot points $\mathbf{c}(u_i), i = 1, \ldots, m+K$. Automatically the system will place the section curves in the 3D space in such a way that the positioning points are matched and each 2D plane on which each section curve lies is aligned to the normal of the spine at that point. Then the section curves can be arbitrarily rotated on their planes until the desired position is obtained.

The swept surface will also interpolate the spine curve whenever the reference point with which the section curve is matched to the spine knot points lies on the section curve itself.

A revolution surface is generated using a NURBS curve which lies in the plane. A full revolution surface is obtained by revolving this curve 2π around the z axis. The resulting surface has the form

$$\mathbf{r}(u,v) = \sum_{i=1}^{9} \sum_{j=1}^{m+K} \mathbf{P}_{ij} R_{i,4}(u) R_{j,m}(v).$$

NURBS swinging is a generalization of the revolution technique, where the trajectory curve is not necessarily circular. Given a profile curve $\mathbf{P}(u)$ in the x, z plane

$$\mathbf{P}(u) = \sum_{i=1}^{p+L} \mathbf{P}_i R_{i,p}(u)$$

with $\mathbf{P}_i = (P_{x_i}, 0, P_{z_i})^T$, and a trajectory curve $\mathbf{T}(v)$ in the x, y plane

$$\mathbf{T}(v) = \sum_{j=1}^{q+L} \mathbf{T}_j R_{j,q}(v)$$

with $\mathbf{T}_j = (T_{x_j}, T_{y_j}, 0)^T$, swinging the profile around the z axis along the trajectory curve yields the surface:

$$\mathbf{r}(u,v) = [sP_x(u)T_x(v), sP_x(u)T_y(v), P_z(u)]$$

with control points and weights:

$$\mathbf{Q}_{ij} = [sP_{x_i}T_{x_j}, sP_{x_i}T_{y_j}, P_{z_i}]^T$$

$$w_{ij} = w_i \cdot w_j,$$

where s is an arbitrary scaling factor.

5.4. – **Object composer**. – Solid constructions can be accomplished by combining, using a set of boolean operations, two or more solids; this process is known as *the set operation algorithm*. Consider, for example, the two solids A and B, respectively defined by their boundary surfaces. The boundaries of the solids $A \bigcup B$, $A \bigcap B$, $A - B$ are determined starting from bA and bB, and trimming away the portion of the patches that does not belong to the resulting solid. The boundary of the resulting solid is given by the equations known as the boundary formula [70]:

$$\begin{aligned} b(A \bigcup B) &= (bA \bigcap cB) \bigcup (bB \bigcap cA) \\ b(A \bigcap B) &= (bA \bigcap iB) \bigcup (bB \bigcap iA) \\ b(A - B) &= (bA \bigcap cB) \bigcup (bB \bigcap iA) \end{aligned}$$

where iX and cX represent, respectively, the interior and the complement of the solid X.

In [13] and [14], a set operation algorithm is given, that operates on solids modelled with trimmed patches. The proposal given in [14] is considered and adapted for NURBS surfaces in [15] and implemented in the *xcmodel* system.

The basic idea in [14] consists in avoiding the intersections between trimmed patches by considering the intersections between the patches over the whole domain. In this way, we can then determine the resulting trimmed regions (2D CSG tree) from the intersection between the trimmed regions of the solids and the trimmed regions obtained by the patch/patch intersection operations.

A detailed description of the set operation algorithm implemented in the system is technically complex. In the following, we will give a brief description of the basic steps involved in this process: the untrimmed surface/surface intersection (SSI), the 2D and 3D Point Membership Classification (PMC), and the 2D curve/curve intersection (CCI).

- SSI -

The SSI procedure implemented in *xcmodel* is a modified version of the proposal given in [3]; details can be found in [17]. The method exploits the advantages offered by the two most frequently used SSI approaches, curve following and subdivision, in order to get a good balance between robustness and efficiency.

The *curve following method* begins by finding some points of intersection. Then, the intersection curve is followed using a numerical method [3, 4]. The *subdivision method* divides the problem into smaller problems by approximating the patch into simpler linear or quadratic sub-patches [48, 14]. The patches are intersected, resulting in curves that are then refined by another numerical method. The proposed modified algorithm roughly proceeds through the four main steps that follow:

- *Adaptive mesh generation*; approximate adaptively each of the surfaces by a grid of isoparametric curves, approximated by piecewise linear curves within a given tolerance. From this grid we easily obtain a surface triangulation;

- *Initial intersection point generation*; compute the intersections between the grid segments of a surface and the triangles of the other surface, in order to obtain at least one starting point for each intersection curve;

- *Following an intersection curve*; starting from an initial point on the intersection curve, move along the curve by steps. This is done by first finding an estimate for the next point on the intersection curve, and then evaluating an exact intersection within a given tolerance;

- *Sorting*; finally, link together the curve intersection segments found in the preceding step in order to obtain close intersection curves.

- 2D and 3D PMC -
PMC is a function that takes, as its input, a point P and a closed set S, and returns one of three possible outcomes: P is inside S, P is on the boundary of S, or P is outside S. The set operation algorithm makes extensive use of PMC in both two and three dimensions. During the construction of the 2D CSG tree resulting from a boolean operation, a single 3D point is classified with respect to the opposing solid. The result will be used for further 2D classifications in order to build a collection of half-spaces which, ultimately, become the leaves of the 2D CSG tree [13, 14].

A common concept is used for both 2D and 3D classification: a ray is extended from the point to be classified, and the number of intersections of the ray with the boundary of the set determines the membership status of the point.

An even number of intersections means that the point is outside the set, while an odd number implies that the point is inside.

While the concept is the same, its implementation is very different for two or three dimensions. In the 2D case, the ray is intersected with polygons or curved boundaries. In the 3D case, the boundary consists of trimmed patches, and a ray/patch method is required [57].

- CCI -
This problem can either be tackled geometrically, that is, using subdivision techniques for the curves, thus exploiting the convex-hull property of the NURBS curves [50, 72], or numerically. In *xcmodel* the geometric approach is implemented.

6. – Scene description

Once the geometric models of the objects have been created, the modelling of a realistic scene is based on positioning, orientating the objects in the scene, characterizing their materials, and defining the light sources illuminating them. This can be realized by applying an illumination model and some textures.
An illumination model is designed to determine the intensity of light reflected by an observer's eye at each point (pixel) in an image. In a global illumination model the intensity of the light reflected from a point to the observer is determined by the light that reaches a point by reflection from, or transmission through, other objects in the scene, as well as the light incident from any light sources.
The illumination model implemented in the system is based on the global illumination model proposed by Whitted [77]. This model derives from the techniques introduced previously by Phong [12] and Blinn [7].

The two traditional methods of texturing implemented in *xcmodel* are texture maps and procedural textures. By a texture mapping, an image is mapped onto an object. This map is realized by positioning the object to be mapped on

the image space and applying an orthogonal projection [7, 38]. A procedural texture is an analytical function defined in 3D space, that is, a function that assigns some visual properties to every point in the 3D texture space. The result of the application of a procedural texture on an object is given by placing the object in the texture space.

7. – Rendering

The rendering of NURBS surfaces as well as curves is a very important task in an interactive graphics system, such as *xcmodel*. We distinguish between low and high level rendering quality.

7.1. – *Low level or real time rendering.* – A low level rendering quality algorithm uses a piecewise planar approximation of the surface in order to produce reasonable images in a reasonable time. *xcmodel* provides two strategies: a uniform approximation and an adaptive approximation within a given tolerance.
The adaptive approximation adopted by the system is not the well known recursive procedure that subdivides the surface until it is "flat enough". In fact this procedure is very time and space consuming. Instead we have implemented the adaptive mesh generation proposed in [3] as the initial step for an SSI method. This consists in a surface approximation using an adaptive grid of isoparametric curves within a given tolerance.
If the surface is trimmed, once the adaptive mesh for the whole surface has been computed, we need to merge the trimming curves and the mesh to yield closed domain polygons and then triangulate them. The triangulation algorithm implemented in *xcmodel* was inspired by [63] and [67]. The latter proposal is not parametrization dependent and proceeds with different tolerances for the trimming curves and the surface. Our implementation differs in the adaptive approximation and in the domain polygon triangulation; in this latter case we adopted a simple but faster heuristic approach than a Delaunay triangulation.
Once a suitable, planar approximation for the surface is obtained, *xcmodel* provides four different low level rendering quality methods:

- a wire frame representation of the grid surface;
- a wire frame representation of the grid surface following the strategy of drawing each grid segment with a different grey level in accordance with the distance from the observer (depth cueing);
- a wire frame representation of the grid surface following the hidden line strategy. The implemented hidden line method was derived from [59] then improved in [53]. An optimization performance strategy was proposed in [16]. This proposal consists of a quasi exact hidden line algorithm highly suitable for a real time visualization.

- a shading algorithm (Gouraud or Phong) of planar approximation. It consists in evaluating the illumination model at various locations: once per vertex for Gouraud shading, and once per pixel for Phong shading [12, 38].

7.2. – *High level or realistic rendering.* – High quality rendering algorithms use an exact representation. *xcmodel* provides a ray tracing algorithm specialized for NURBS surfaces. The basic problem in a ray-tracing surface algorithm is the ray/patch intersection. If the surface is trimmed, it follows a step to determine whether the intersection point lies inside or outside a trimmed region using a 2D PMC algorithm. In *xcmodel* three methods for ray/patch intersection are implemented:

- Toth [76]
  Toth's algorithm is based on interval Newton iteration. It works robustly on any parametric surface for which bounds onto the surface and its first derivative can be obtained.

- Bezier Clipping [57]
  This algorithm uses the convex-hull property in a powerful manner, by determining parameter ranges which guarantee that they do not include points of intersection. Bezier Clipping has the flavor of a geometrically based interval Newton method, and thus may be categorized as partly a subdivision based algorithm and partly a numerical method.

- Toth speed [56, 74]
  A new approach, following one of our ideas, was called Toth speed. It is a combination of the Toth and Bezier Clipping algorithms that outperforms their performances. The idea is to reduce the application of the interval Newton iterations of the Toth method, trying to find the solution using a simple Newton, in the knowledge that, if a solution exists, it is unique. If this fails, an interval Newton iteration is applied. Moreover, when the Toth algorithm is forced to use binary subdivision, our method uses Bezier Clipping.

Our ray tracing implementation exploits a certain number of optimization techniques [42] in order to speed up performance, such as subdividing a 3D scene into a uniform grid of voxels [41], subdividing the NURBS surfaces in rational Bezier patches, further subdividing each rational Bezier patch until a given flat tolerance is reached and introducing a second level of subdivision of the surface bounding boxes [56].

In Fig.3, some modelling examples produced using *xcmodel* system by undergraduates in Computer Science at the University of Bologna, are shown.

Figure 3: Scenes modelled and rendered with the *xcmodel* system

8. – Acknowledgements

The system has developed over several years, and thanks are due to a number of people who made significant and minor contributions to its development and testing.

REFERENCES

[1] I. AKIMA, *A new method of interpolation and smooth curve fitting based on local procedures*, Journal of ACM, **17** (1970), pp. 589–602.

[2] L. ALT, *Rational linear reparametrization of NURBS and the blossoming principle*, Computer Aided Geometric Design, **10** (1993), pp. 465–467.

[3] R.E. BARNHILL - G. FARIN - M. JORDAN - B.R. PIPER, *Surface Surface Intersection*, Computer Aided Geometric Design, **4** (1987), pp. 3–16.

[4] R.E. BARNHILL - S.N. KERSEY, *A marching method for parametric surface/surface intersection*, Computer Aided Geometric Design, **7** (1990), pp. 257–280.

[5] B.A. BARSKY - D.P. GREENBERG, *Determining a set of B-spline control vertices to generate an interpolation surface*, Computer Graphics and Image Processing, **14** (1980), pp. 203–226.

[6] R.H. BARTELS - J.C. BEATTY - B.A. BARSKY, *An introduction to splines for use in computer graphics and geometric modelling*, Morgan Kaufman publishers, (1987).

[7] J.F. BLINN - M.E. NEWELL, *Texture and reflection in computer generated images*, Communication of ACM, **19** (1976), pp. 542–547.

[8] W. BOEHM, *Inserting new knots into B-spline curves*, Computer Aided Design, **12** (1980), pp. 199–201.

[9] W. BOEHM, *On the efficiency of knot insertion algorithms*, Computer Aided Geometric Design, **2** (1985) pp. 141–143.

[10] W. BOEHM - H. PRAUTZSCH, *The insertion algorithm*, Computer-Aided Design, **17** (1985), pp. 58–59.

[11] C. DEBOOR, *On calculating with B-splines*, Journal of Approximation Theory, **6** (1972), pp. 50–62.

[12] BUI-TUONG - PHONG, *Illumination for computer generated pictures*, Communication of ACM, **18** (1975), pp. 449–455.

[13] M.S. CASALE, *Free-form solid modeling with trimmed surface patches*, IEEE Computer Graphics & Applications, **Jan.** (1987), pp. 33–43.

[14] M.S. CASALE - J.E. BOBROW, *A set operation algorithm for sculptured solids modeled with trimmed patches*, Computer Aided Geometric Design, **6** (1989), pp. 235–247.

[15] G. CASCIOLA - B. QUAQUARELLI, *Primitive solide, operazioni Booleane e NURBS*, Proceedings of ICO-GRAPHICS'90, (1990).

[16] G. CASCIOLA - S. MORIGI, *Graphics in parallel computation for rendering 3D modelled scenes*, Parallel Computing, **21** (1994), pp. 1365–1382.

[17] G. CASCIOLA - S. MORIGI, *Il problema SSI nella modellazione solida con superfici NURBS*, Atti dell'Accademia delle Scienze dell'Istituto di Bologna, **Serie V** (1995), pp. 107–127.

[18] G. CASCIOLA - S. MORIGI, *Reparametrization of NURBS curves*, International Journal Shape Modelling, **2** (1996), pp. 103–116.

[19] G. CASCIOLA - S. MORIGI, *The trimmed NURBS age*, submitted, (1999).

[20] G. CASCIOLA, *xcmodel: a system to model and render NURBS curves and surfaces, User's Guide - Version 1.0*, Progetto MURST: "Analisi Numerica: Metodi e Software Matematico", Ferrara (2000)
http://www.dm.unibo.it/∼casciola/html/xcmodel.html

[21] G. CASCIOLA, *xccurv: the 2D modeller, User's Guide - Version1.0*, Progetto MURST: "Analisi Numerica: Metodi e Software Matematico", Ferrara (2000)
http://www.dm.unibo.it/∼casciola/html/xcmodel.html

[22] G. CASCIOLA, *xcsurf: the 3D modeller, User's Guide - Version 1.0*, Progetto MURST: "Analisi Numerica: Metodi e Software Matematico", Ferrara (2000)
http://www.dm.unibo.it/∼casciola/html/xcmodel.html

[23] G. CASCIOLA - G. DEMARCO, *xcbool: the object composer, User's Guide - Version 1.0*, Progetto MURST: "Analisi Numerica: Metodi e Software Matematico", Ferrara (2000)
http://www.dm.unibo.it/∼casciola/html/xcmodel.html

[24] G. CASCIOLA, *xcrayt: the scene descriptor, User's Guide - Version 1.0*, Progetto MURST: "Analisi Numerica: Metodi e Software Matematico", Ferrara (2000)
http://www.dm.unibo.it/∼casciola/html/xcmodel.html

[25] G. Casciola, *MATRIX library: Programming Guide - Version 1.0*, (1999) http://www.dm.unibo.it/∼casciola/html/xcmodel.html

[26] G. Casciola - S. Bonetti, *descriptor library: Programming Guide - Version 1.0*, (1999) http://www.dm.unibo.it/∼casciola/html/xcmodel.html

[27] G. Casciola - G. DeMarco, *trim library: Programming Guide - Version 1.0*, (1999) http://www.dm.unibo.it/∼casciola/html/xcmodel.html

[28] G. Casciola - S. Bonetti, *xtools library: Programming Guide - Version 1.0*, (1999) http://www.dm.unibo.it/∼casciola/html/xcmodel.html

[29] E. Cohen - T. Lyche - R.F. Riesenfeld, *Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics*, Computer Graphics and Image Processing, **14** (1980), pp. 87–111.

[30] E. Cohen - T. Lyche - L.L. Schumaker, *Algorithms for degree-raising of splines*, Transaction On Graphics, **4** (1985), pp. 171–181.

[31] E. Cohen - T. Lyche - K. Morken, *Knot line refinement algorithms for tensor product B-splines surfaces*, Computer Aided Geometric Design, **2** (1985), pp. 133–139.

[32] E. Cohen - T. Lyche - L.L. Schumaker, *Degree raising for splines*, Journal of Approximation Theory, **46** (1986), pp. 170–181.

[33] M.G. Cox, *The numerical evaluation of B-splines*, J. Inst. Math. Appl., **10** (1972), pp. 134–149.

[34] T. Dokken, *Finding intersections of B-spline represented geometries using recursive subdivision techniques*, Computer Aided Geometric Design, **2** (1985), pp. 189–195.

[35] G. Farin, *Curves and surfaces for CAGD: a practical guide*, Academic Press Inc., (1993).

[36] R.T. Farouki, *Trimmed-surface algorithms for the evaluation and interrogation of solid boundary representations*, IBM J.RES.DEVELOP., **31** (1987), pp. 314–333.

[37] R.T. Farouki - T. Sakkalis, *Real rational curves are not 'unit speed'*, Computer Aided Geometric Design, **8** (1991), pp. 151–157.

[38] J.D. Foley - A. VanDam - S.K. Feiner - J.F. Hughes, *Computer Graphics principles and practice, II Edition*, Addison Wesley, (1990).

[39] D.R. Forsey - R.H. Bartels, *Hierarchical B-spline refinement*, Proceeding SIGGRAPH'88, In Computer Graphics, **22** (1988), pp. 205–212.

[40] R.D. FUHR - M. KALLAY, *Monotone linear rational spline interpolation*, Computer Aided Geometric Design, **9** (1992), pp. 313–319.

[41] A. FUJIMOTO - T. TANAKA - K. IWATA, *ARTS: accelerated ray-tracing system*, IEEE Comp. Graphics and Appl., **April** (1986), pp. 16–26.

[42] A. GLASSNER ED., *Introduction to ray tracing*, Academic Press, (1989).

[43] A. GRANDINE, *Computing zeroes of spline functions*, Computer-Aided Geometric Design, **6** (1989), pp. 129–136.

[44] J.A. GREGORY - M. SARFRAZ, *A rational cubic spline with tension*, Computer Aided Geometric Design, **7** (1990), pp. 1–13.

[45] CH.M. HOFFMANN, *Geometric and Solid Modeling. An Introduction*, Morgan Kaufmann Publishers, (1989).

[46] M.E. HOHMEYER - B.A. BARSKY, *Rational continuity: parametric, geometric and frenet frame continuity of rational curves*, ACM Transaction on Computer Graphics, **8** (1989), pp. 335–359.

[47] M.E. HOHMEYER - B.A. BARSKY, *Skinning rational B-spline curves to construct an interpolatory surface*, CVGIP: Graphical Models and Image Processing, **53** (1991), pp. 511–521.

[48] E.G. HOUGHTON - R.F. EMNETT - J.D. FACTOR - C.L. SABHARWAL, *Implementation of a divide-and-conquer method for intersection of parametric surfaces*, in R.E.Barnhill, W.Boehem, eds. Surfaces in Computer Aided Geometric Design, North-Holland, (1984).

[49] P. LANCASTER - K. SALKAUSKAS, *Curve and surface fitting, an introduction*, Academic Press, (1986).

[50] J.M. LANE - R.F. RIESENFELD, *A theoretical development for the computer generation and display of piecewise polynomial surfaces*, IEEE Transaction on PAMI, **2** (1980), pp. 35–46.

[51] E.T.Y. LEE, *Choosing nodes in parametric curve interpolation*, Computer-Aided Design, **21** (1989), pp. 363–370.

[52] E.T.Y. LEE - M.L. LUCIEN, *Möbius reparametrization of rational B-splines*, Computer Aided Geometric Design, **8** (1991), pp. 213–215.

[53] L. LI, *Hidden-line algorithm for curved surfaces*, Computer-Aided Design, **20** (1988), pp. 466–470.

[54] T. LYCHE - K. MORKEN, *Making the Oslo algorithm more efficient*, SIAM J. Numer. Anal., **23** (1986), pp. 663–675.

[55] W. MA - J.P. Kruth, *Parametrization of randomly measured points for least square fitting of B-spline curves and surfaces*, Computer-Aided Design, **27** (1995), pp. 663–675.

[56] M. Marini - D. Grilli, *Un algoritmo di ray-tracing per superfici spline razionali trimmate*, Master Thesis in Computer Science, University of Bologna, (1995).

[57] T. Nishita - T.W. Sederberg - M. Kakimoto, *Ray tracing trimmed rational surface patches*, ACM Computer Graphics, **24** (1990), pp. 337–345.

[58] G. Nurnberger, *Approximation by Spline Functions*, Springer Verlag, (1989).

[59] Y. Ohno, *A hidden line elimination method for curved surfaces*, Computer-Aided Design, **15** (1983), pp. 209–216.

[60] L.A. Piegl, *Modifying the shape of rational B-splines. Part 1: curves*, Computer-Aided Design, **21** (1989), pp. 509–518.

[61] L.A. Piegl, *Modifying the shape of rational B-splines. Part 2: surfaces*, Computer-Aided Design, **21** (1989), pp. 538–546.

[62] L.A. Piegl, *On NURBS: a survey*, IEEE Comput. Graph. and Appl., **10** (1991), pp. 55–71.

[63] L.A. Piegl - A.M. Richard, *Tessellating trimmed NURBS surfaces*, Computer-Aided Design, **27** (1995), pp. 16–26.

[64] L.A. Piegl - W. Tiller, *Software-engineering approach to degree elevation of B-spline curves*, Computer-Aided Design, **26** (1994), pp. 17–28.

[65] L.A. Piegl - W. Tiller, *Algorithm for approximate NURBS skinning*, Computer-Aided Design, **28** (1996), pp. 699–706.

[66] L.A. Piegl - W. Tiller, *The NURBS book*, Springer, (1995).

[67] L.A. Piegl - W. Tiller, *Geometry-based triangulation of trimmed NURBS surfaces*, Computer-Aided Design, **30** (1998), pp. 11–18.

[68] H. Prautzsch, *Degree elevation of B-spline curves*, Computer Aided Geometric Design, **1** (1984), pp. 193–198.

[69] H. Prautzsch - B. Piper, *A fast algorithm to raise the degree of spline curves*, Computer Aided Geometric Design, **8** (1991), pp. 253–265.

[70] A.A.G.Requicha, *Mathematical models of rigid solids*, Tech. Memo. 28, Production Automation Project, University of Rocheste (1977).

[71] S. ROTH, *Ray casting for solid modeling*, Computer Graphics and Image Processing, **18** (1982), pp. 109–144.

[72] T.W. SEDERBERG - T. NISHITA, *Curve intersection using Bezier-clipping*, Computer-Aided Design, **24** (1990), pp. 538–549.

[73] T.W. SEDERBERG - S.R. PARRY, *Comparison on three curve intersection algorithm*, Computer-Aided Design, **18** (1986), pp. 58–63.

[74] S. SPAGNA, *Analisi, sviluppo e applicazione di metodi di intersezione per spline razionali*, Master Thesis in Compuer Science, University of Bologna, (1998).

[75] W. TILLER, *Knot-removal algorithms for NURBS curves and surfaces*, Computer-Aided Design, **24** (1992), pp. 445–453.

[76] D. TOTH, *On ray tracing parametric surfaces*, Proceeding of SIGGRAPH'85, In Computer Graphics, **19** (1985), pp. 171–179.

[77] T. WHITTED, *An improved illumination model for shaded display*, Communication ACM, **23** (1980), pp. 96–102.

[78] C.D. WOODWARD, *Cross-sectional design of B-spline surfaces*, Computer & Graphics, **11** (1987), pp. 193–201.

[79] C.D. WOODWARD, *Skinning techniques for interactive B-spline surface interpolation*, Computer-Aided Design, **20** (1988), pp. 441–451.