Abstract: - Generating a Surface of a desired shape as a single patch is computationally complicated and inefficient. We need to use simple small patches approximating small portions of the given/probable surface. For generating the surface as a patchwork of $n \times m$ array of Bicubic Bézier surface patches we need to have $4(n + m) + 8mn$ no of control points(vectors). We present here design and implementation of our interactive software tool, called BezierPkg, which takes $nm$ input vectors to generate a smooth patchwork of freeform surfaces that interpolate the given set of points. We have built BezierPkg as an external module to the well known publicly available geometric viewer Geomview. We also present an outline of new algorithms for smoothening a patchwork of surfaces and for convexity related problems.

Key-Words: - CAD, CAGD, Bézier curves and surfaces, Solid Modeling, Geomview, Visualization

1 Introduction

Computer Aided Geometric Design (CAGD) is a branch of Mathematics and Computer Science dealing with freeform curves and surfaces and their applications in engineering [4]. Development of tools to support CAGD is always of relevance. In particular, creating smooth patchwork of parameterized surfaces is of great importance. We present, in the paper, BezierPkg, an interactive tool developed by us, that can be used to produce smooth patchwork of surfaces. This tool facilitates precise user interaction and incorporates algorithms for creating and modifying the patchwork of surface. Inspite of a large number of tools available for interactive viewing and rendering, relatively fewer inexpensive tools are available that facilitates generation of smooth patchwork of surfaces. We endeavoured to build one such with limited but useful functionality.

BezierPkg is built to function as an external module to the well known public domain geometric viewer tool called Geomview [5]. It augments the functionality, to Geomview, for interactive creation and manipulation of the CAGD geometric objects (Geomview supports rendering of Bézier surfaces, however not modifications of them). BezierPkg provides the necessary graphical user interface (GUI) to augment the GUI facilities available in Geomview. In addition, more critically, BezierPkg incorporates new algorithms based on CAGD principles that allow the user to create smooth patchwork of surfaces. Several external modules have been developed on top of Geomview [6, 7], however very little appears to have been done to build an external module to Geomview for CAGD research. Furthermore, typical solid modeler usually provides the following: 1) creation and modification of 3d geometries using solid modeling primitives such as sweeping, stitching, extrusion etc. 2) facilitating simple local modifications at picked geometric entities. Techniques employed by BezierPkg falls somewhere in the middle of the above two categories: Simple as well as global! However, we pay the price for the simplicity and globalness by limiting ourselves to dealing with closed patchwork of quadrilateral surfaces.

We have used our tool to investigate complex con-
vexification techniques and aim to use it for grid
generation of 3-d volumes as well as for fabrication
of moulds. The geometries produced by BezierPkg
consist of Bézier surface patches and can therefore
be easily passed over to more powerful, expensive
standard solid modellers for more complex opera-
tions. Indeed we are ourselves engaged in develop-
ing ACIS [1] based backend for complex geometric
operations on the initial geometries created by
BezierPkg. A final remark is that our tool is very
compact piece of software with approx. 4000 lines of
Java code which makes it easy to maintain and scale.
The integration with Geomview has enabled us to
offload the responsibility of rendering and viewing
operations to Geomview thereby allowing us to fo-
cus on the algorithms and the necessary user inter-
face development to augment to the features of Ge-
geomview.

The roadmap to this paper is as follows: In the
next section we present a few relevant definitions and
concepts. In section 3 we briefly describe the pack-
age Geomview [5]. In section 4 an outline of the
design and the functionality of our tool BezierPkg is
presented. Next in section 5 a few of the algorithms
designed and implemented for the purpose of gener-
ation of smooth patchwork of surfaces are presented.

2 Preliminaries: Bézier Curves and
Surfaces

A parameterized curve \( c(t) \) is a map \( c : I \rightarrow R^3 \),
where \( I \) is an interval on real line. A curve \( c(t) \) is
said to be \( C^k \) at \( t = \alpha \) if it is \( k \) times differentiable at
\( t = \alpha \). A curve \( c(t) \) is said to be \( G^1 \) at \( t = \alpha \) if its left
and right derivatives at \( t = \alpha \) are scalar multiples of
each other. The \( i^{th} \) Bernstein polynomial of degree
\( n \) is given by:

\[
B_i^n(t) = \binom{n}{i} (1 - t)^{n-i} t^i
\]

A Bézier curve is a parameterized curve defined as
follows:

\[
\text{Bez}(P_0, P_1, \ldots, P_n)(t) = \sum_{i=0}^{n} P_i B_i^n(t) \quad 0 \leq t \leq 1
\]

where the \( B_i^n \)'s are given by Eq 1, and \( P_i \)'s are points
in 3-d space. These points are called control points
as they define the curve completely. The sequence of
these control points is called control polygon.

Bézier curves and surfaces are extremely useful
due to their fundamental properties which include
the easy characterization of their smoothness. Indeed
their smoothness can be characterized using the con-
trol points, which helps a lot in designing algorithms
for ensuring smoothness of various degree.

A Cartesian or tensor product Bézier surface is
given by

\[
s(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{ij} B_i^n(u) B_j^m(v)
\]

where \( B_i^n \) and \( B_j^m \) are the Bernstein basis functions
in the \( u \) and \( v \) parametric directions and defined as
in Equation 1. \( P_{ij}, 0 \leq i \leq n, 0 \leq j \leq m \) are
called control points of the Bézier surface. These
points together are referred to as control mesh. We
now present some more preliminaries about Bézier
curves and surfaces. For details the reader is referred
[4].

In this paper, we deal almost exclusively with
bicubic Bézier surface patches which are simply the
Bézier surfaces of degree 3 each in \( u \) and \( v \) direc-
tion. Thus a bicubic Bézier surface patch is de-
scribed by \( 4 \times 4 \) mesh of control points. We say that
inner control points of a bicubic Bézier patch satisfy
paralellogram law if \( P_{1, 1} - P_{1, 0} = P_{0, 1} - P_{0, 0} \),
\( P_{1, 2} - P_{1, 3} = P_{0, 2} - P_{0, 3} \), \( P_{2, 1} - P_{2, 0} = P_{3, 1} - P_{3, 0} \)
and \( P_{2, 2} - P_{2, 3} = P_{3, 2} - P_{3, 3} \).

For two adjacent bicubic Bézier surfaces \( s_1(u, v) \)
and \( s_2(u, v) \) with common boundary curve as
\( s_1(u, 1) \) (or equivalently \( s_2(u, 0) \)), the cross bound-
ary \( C^1 \) continuity condition (along \( v \) direction) is
\[
\frac{\partial}{\partial u} s_1(u, v) \big|_{v=1} = \frac{\partial}{\partial u} s_2(u, v) \big|_{v=0} \quad \text{and cross boundary}
\]
\( C^2 \) continuity conditions (along \( v \) direction) are
\[
\frac{\partial^2}{\partial u^2} s_1(u, v) \big|_{v=1} = \frac{\partial^2}{\partial u^2} s_2(u, v) \big|_{v=0} \quad \text{and } \frac{\partial^2}{\partial u \partial v} s_1(u, v) \big|_{v=1} = \frac{\partial^2}{\partial u \partial v} s_2(u, v) \big|_{v=0} \quad \text{for } u \in [0, 1].
\]
We have similar continuity conditions along \( u \) direction also.

Note that if the inner control points of a bicubic
Bézier surface satisfy paralellogram law, the twist
vectors of the surface i.e., the values of \( \frac{\partial}{\partial u} \) at the
corner points will be zero. We used this condition
in our algorithm explained in subsection 5.2.
3 Geomview

Geomview [5], developed by Geometry Center, University of Minnesota, is a widely used powerful interactive program for viewing and manipulating geometric objects. Geomview’s main purpose is to display objects whose geometry is given, allowing interactive control over details such as point of view, speed of movement, appearance of surfaces and lines, and so on. Geomview can handle any number of objects and allows both separate and collective control over them.

Geomview supports many of the data types. Bézier surface patches of arbitrary degree is one of them.

We can also use Geomview to handle the display of data coming from another program that is running simultaneously. As the other program changes the data, the Geomview image reflects the changes. Programs that generate objects and use Geomview to display them are called external modules. External modules can control almost all aspects of Geomview. The idea here is that many aspects of the display and interaction parts of geometry software are independent of the geometric content and can be collected together in a single piece of software that can be used in a wide variety of situations. The author of the external module can then concentrate on implementing the desired algorithms and leave the display aspects to Geomview.

4 Architecture of BezierPkg

The design of our package BezierPkg provides an “easy to use” GUI to interactively develop a surface from a grid of points. The input and output software are in ”.patch” and ”.srf” files which are in the text format and can easily understood although the surface patchwork itself can be of complicated shape. This allows us to integrate BezierPkg with any software which has ability to process bicubic Bézier patches.

The .patch files describe a collection of quadrilaterals (each quadrilateral is described by its four corner points) and their adjacency information. The .srf file contains the information about the smoothened patchwork of bicubic Bézier surface patches that are produced by our software. Files of either kind can be produced by hand or by software tools. We use our external module Cuboid to produce the .patch files and use the external module Bezier to optionally read in such quadrilateral descriptions and allow the user to generate smoothened patchwork of bicubic Bézier surfaces and store them in .srf files.

The surface patches or quadrilaterals are generated by very simple user interaction of picking points and modifying them precisely. Most other tools either allow very powerful and difficult GUI for doing this or allow simple user interface but for local shape modifications only. The algorithms in BezierPkg allow more global changes as well as they maintain the common boundaries. For example, in the model of the chair, the sides are patchworks of planar Bézier surfaces and have been automatically generated. Thus the sides can also be modified easily without affecting the chair seat etc. In essence, simple interaction results in a large number of bicubic Bézier surfaces that patch with each other well. It should be noted that this is not the same as simply generating sweep surface or any other straightforward application of solid modeling primitives.

We are also engaged in making a port of this using the powerful, industry standard ACIS geometry kernel [1] for developing a surface modeler to take care of various subtleties in surface design. BezierPkg consists of the following conceptual components:

Database It consists of
• collection of 3-d quadrilaterals stored in files with “.patch” extension
• control points of rectangular array of surfaces stored in files with “.srf” extension

Functionality It provides the following functionalities during the interactive session
• Creating a patchwork from a file with “.patch” extension
• Displaying and manipulating a patchwork of rectangular array of bicubic Bézier surfaces from a file of “.srf” extension.
• Creating and manipulating a default surface patchwork of given no. of rows and columns. The default grid is on a spherical cap.
• Creating and manipulating a default closed surface starting from a cuboid of specified order along length, width and height.
• Storing the manipulated surface patchwork as files with “.srf” extension.
Next we describe the two external modules (to Geomview) which make up the BezierPkg.

4.1 External Module: Bézier

The user is given options to Create a Default Mesh of a specified size or Choose Patches from Database or Read the surface from Database.

In the case of the first option of creating a default mesh of points (which by default will lie on a spherical cap) however these points can be modified interactively.

Using the option Choose patches a user may load the collection of 3-d quadrilaterals from a file. The user can then iterate through these quadrilateral patches to get familiar with them, use Geomview to study the patches from different views. The user then can select a few of these patches and choose options to smoothen them together creating a smooth patchwork of bicubic Bézier surfaces.

With the help of the option Read Surface From Database a file (with .srf extension) can be loaded in the viewer. Such a file already contains a collection of Bézier patches, however these surfaces can also be modified interactively.

4.2 External Module: Cuboid

This module allows the user to begin with a initial cuboid with the specified height, width and length and the breakups in each dimension. By virtue of the facility of modifying the grid points on the Cuboid interactively, a good piecewise linear initial approximation can be created for a closed smooth patchwork of freeform surfaces. The output of this module is essentially a collection of 3-d quadrilaterals.

Figure 1: Model of a mouse generated with the help of BezierPkg

Figure 2: Model of a chair generated with the help of BezierPkg

5 Algorithms in BezierPkg

We shall now outline the main algorithms used in creating smooth patchwork of bicubic Bézier patches with minimal and simple user interaction. It may be noted that our basic user interaction is simply that of picking points and modifying their locations. As compared to most other tools, our algorithms focus on maintaining smoothness more globally.

5.1 $C^1$ interpolation of points

Here we will discuss the interpolation of a given set of points with the help of patchwork $c(t)$ of Bézier curve segments(between each pair of adjacent points).

**Cubic Hermite Interpolation** Hermite interpolation is a generalization of Lagrange interpolation [4].
boundary control points for each surface patch.

- We get inner control points of each bicubic Bézier patch using parallelogram law.

In the patchwork computed above, though cross boundary $C^2$ continuity may not always be possible at each point on the boundary curve of adjacent patches but it satisfies much stronger condition than $C^1$ continuity as evident from the lemmas stated below.

**Lemma:** The patchwork of Bézier surfaces computed above interpolates the rectangular grid of points $C^2$ continuously in the sense that at each grid point (corner points of patches) the patchwork is $C^2$ continuous.

**Lemma:** For each pair of adjacent patches in the patchwork computed above $C^1$ continuity condition is satisfied at each of their boundary points.

### 5.3 $C^1$ continuous interpolation of points

$C^1$ interpolation of a sequence of points $\{q_i\}_{i=0}^n$ by a patchwork of cubic Bézier curves is such that at the intermediate points $\{q_i\}_{i=1}^{n-1}$ right hand derivative is a scalar multiple (say $k_i$) of the left hand derivative so that the curve is visually smooth at those points. Once the unit slope vectors at the intermediate points are fixed this condition gives the control points adjacent to them of the cubic Bézier curves. We take the unit slope vectors to be the unit vectors along the left adjacent point to the right adjacent point at each intermediate points. We need to compute control points adjacent to the end points. For doing so at $q_0$ we first find the angle $\alpha$ the line $l_1$, parallel to the slope vector at $q_1$ and passing through it, makes with the line $l$, between the points $q_0$ and $q_1$. We compute the line $l_2$ making angle $\alpha$ with $l$. We take the required control point to be the midpoint of $q_0$ and the intersection point of $l_1$ and $l_2$. Similarly we compute the control point adjacent to $q_n$.

### 5.4 $C^1$ Spline surfaces

We compute $4 \times 4$ mesh of control points for each surface patch to get a bicubic Bézier surface representation of it. That is we need to have $(3n-2)(3m-2)$ no. of points from the given $nm$ points. Mesh of

Figure 3: Model of a car generated with BezierPkg

Let function values at the abscissae values $x_0 = 0$ and $x_1 = 1$ be $f_0$ and $f_1$, and slopes be $f'_0$ and $f'_1$. The cubic polynomial that interpolates the function such that it has same derivative values as the function at the abscissae values is given by $H_0(x)f_0 + H_1(x)f'_0 + H_2(x)f'_1 + H_3(x)f_1$ where $H_0(x) = 1 - 3x^2 + 2x^3$, $H_1(x) = 3x^2 - 2x^3$, $H_2(x) = x - 2x^2 + x^3$ and $H_3(x) = -x^2 + x^3$. $H_i(x)$ are called cubic Hermite polynomials.

#### 5.2 Algorithm for $C^1$ continuous patchwork of bicubic Bézier patches

Consider $(n + 1) \times (m + 1)$ array of points $\{P_{i,j}\}$. We have to compute control points of $n \times m$ array of bicubic Bézier surfaces whose patchwork interpolates the grid of points $\{P_{i,j}\}$. For each column of points $\{P_{i,j}\}_{i=0}^n$ we choose the slope vectors at $P_{0,j}$ as $2P_{0,j} + P_{1,j}$ and slope vectors at the last point $P_{n,j}$ as $2P_{n,j} + P_{n-1,j}$. Similarly for each row of points $\{P_{i,j}\}_{j=0}^m$ we choose the slope vectors at $P_{i,0}$ as $2P_{i,0} + P_{i,1}$ and slope vectors at the last point $P_{i,m}$ as $2P_{i,m} + P_{i,m-1}$. After fixing the slope vectors at the end points we go through following steps:

- For each column of point (along with their slope vectors) use Hermite interpolation to get $C^2$ patchwork of Bézier curves.
- Similarly get the $C^2$ patchwork of Bézier curves for each row of points. Thus we get the

...
control points calculated for each patch is such that the first order partial derivatives along the boundary lines of two adjacent patches are proportional. The algorithm is described below:

**Step 1:** For each row and column of points we compute the $G^1$ interpolant curve as described above. Thus we get boundary control points for each surface patch.

**Step 2:** In this step we use parallelogram law to calculate inner four control points for each Bézier surface patch. The three points of the parallelogram is the given corner point of the surface patch and its two adjacent points calculated in the previous steps.

Note that parallelogram law for the inner control points ensures cross boundary $G^1$ continuity of the patchwork if for each column of points the ratio of left hand derivative and right hand derivative along row direction must be equal and also for each row of points the ratio of left hand derivative and right hand derivative along column direction must be equal. They should be chosen appropriately in the first step.

### 5.5 Convexification of surfaces

Convex surfaces play an important role in surface designing. A surface is convex if its Gaussian curvature (discussed below) is non-negative everywhere. We have implemented algorithms which can compute discretized mesh surface of numerator part of a Bézier patch and display it with the help of Geomview renderer along with its surface. We have also developed a software which computes an almost smooth convex patchwork surface of bicubic Bézier patches. The patchwork best fits the given rectangular grid of points.

#### 5.5.1 Gaussian curvature and convexity of surfaces

Gaussian curvature at a point on a surface $s(u, v)$ is the product of maximum and minimum values of curvatures of curves at the point which implies that positivity of Gaussian curvature is a sufficient condition for the surface to be convex at the point. It is given by $\kappa(u, v) = \kappa_{num}(u, v)/\kappa_{den}(u, v)$ where $\kappa_{den}(u, v)$ is positive for all values of $u, v$ and $\kappa_{num}(u, v)$ is a polynomial of degree 14 for a bicubic Bézier patch. The Gaussian curvature at a point is positive if its numerator part $\kappa_{num}(u, v)$ is positive. Analysis of a high order polynomials are intricate and computationally difficult. We therefore take the uniform discretization of the functional surface of the Gaussian curvature function for the analysis of the convexity of the bicubic Bézier patch. For the purpose of searching a convex surface around a given non-convex surface we look for control points in each iteration which increases the minimum of the Gaussian curvature values in the discretized domain. We use Hooke’s and Jeeves pattern search method as a heuristic search for this purpose. The Hooke’s and Jeeves pattern search method is an optimization technique and we use it to iteratively maximise our objective function which is minimum of the Gaussian curvature values in the discretized domain. We know that the Gaussian curvature is an unbounded function and we use the optimising technique to increase the value of the objective function till it becomes positive.

![Figure 4: Non-convex surface along with magnified view of the Gaussian curvature plot in the discretized domain. The Gaussian curvature plot going below the $XY$ plane says that the surface is non-convex](image)

### References

Figure 5: Convex surface generated by our software from the non-convex surface shown in Figure 4


