Computer-Aided Design 71 (2016) 28–38

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Can local NURBS refinement be achieved by modifying only the user interface?☆

Neil A. Dodgson *, Jiří Kosinka

The Computer Laboratory, University of Cambridge, 15 J. J. Thomson Avenue, Cambridge, CB3 0FD, UK

ARTICLE INFO

Article history:
Received 12 February 2015
Accepted 15 September 2015

Keywords:
NURBS
User interface
Local refinement
Surface design
Hierarchical B-spline
Control mesh

ABSTRACT

NURBS patches have a serious restriction: they are constrained to a strict rectangular topology. This means that a request to insert a single new control point will cause a row of control points to appear across the NURBS patch, a global refinement of control. We investigate a method that can hide unwanted control points from the user so that the user's interaction is with local, rather than global, refinement. Our method requires only straightforward modification of the user interface and the data structures that represent the control mesh, making it simpler than alternatives that use hierarchical or T-constructions. Our results show that our method is effective in many cases but has limitations where inserting a single new control point in certain cases will still cause a cascade of new control points to appear across the NURBS patch.

1. Introduction

NURBS are the standard mechanism for modelling in CAD. For decades [1], there has been interest in producing hierarchical NURBS, NURBS with T-junctions, and other NURBS variants that allow for local refinement of a NURBS patch (Section 3). None of these solutions, however, has yet been widely adopted in the CAD industry. Some require significant changes to the underlying NURBS engine. We investigate whether it is possible to construct a mechanism that provides local refinement to the designer by modifying *only* the user interface, leaving the underlying NURBS engine unchanged (Sections 5 and 6).

Our motivation is that providing local refinement through the user interface alone would allow CAD software providers to add the extra functionality without the need to make expensive additions and changes to the underlying NURBS engine. Our investigation shows that our method does deliver such functionality but that it suffers from inescapable limitations (Section 8). Nevertheless, this idea provides an interesting intermediate option between the status quo and adoption of a new engine.

2. The challenge

Bivariate NURBS patches are composed, in parameter space, as the tensor product of univariate NURBS. It is well known that, in the univariate case, a NURBS curve can be locally refined arbitrarily often in arbitrary locations (Fig. 3). A NURBS patch cannot be refined arbitrarily often at arbitrary point locations, owing to its tensor product nature. Any refinement of the NURBS patch will stretch from one side of the patch to the other (Fig. 1).

Our basic idea is to provide a mechanism whose user interface shows only the desired control points to the designer. That is, it hides unwanted control points. We implement this as a series of tensor product control meshes, each of which we call a *layer*. Each layer is a refinement of the layer above in which a single knot is added. Some points from a given layer may be visible to the user and some may be hidden. The rationale here is that the positions of the hidden control points, in the refined layer, can be calculated from control points in the previous layer without altering the shape of the surface. This is just basic knot insertion where, in the univariate case, inserting a single knot in a curve of order k (degree $k-1$) causes one new control point to be introduced and $k-2$ existing control points to be moved without changing the shape of the curve.

Our basic idea is illustrated in Fig. 2. Fig. 2(a) shows what the user sees in the user-interface. Fig. 2(b)–(f) shows how this can be implemented as a series of layers, each of which introduces a single new knot. The bottom-most layer, Fig. 2(f), is a tensor-product NURBS that is passed to the underlying NURBS engine. There are three types of points: *visible* control points available in the user interface (coloured circles), *replaced* control points (grey circles) that have been superseded by points in a lower layer, and *hidden* points (coloured diamonds) that are calculated from points in the layer above. Note that every layer is a tensor-product arrangement,

☆ This paper has been recommended for acceptance by Tae-wan Kim.

* Corresponding author. Tel.: +44 1223 334417; fax: +44 1223 334678.
E-mail address: nad@cl.cam.ac.uk (N.A. Dodgson).

http://dx.doi.org/10.1016/j.cad.2015.09.006

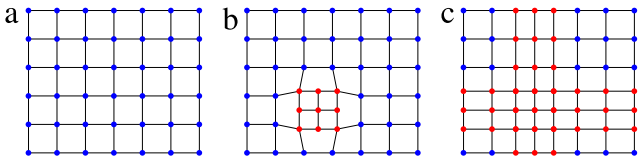

Fig. 1. (a) A NURBS patch showing control points in a regular grid. (b) The desirable approach to local refinement, with new control points in a local area only. (c) Attempting to do this with NURBS introduces new control points across the mesh owing to the tensor product nature of NURBS patches.

while the control mesh visible to the user is not necessarily tensor-product and is constructed by building a mesh from the control points that are marked as *visible* in the various layers.

3. Related work

Since their invention in the 1970s, NURBS [2], a non-uniform rational extension of B-splines, have become a universal standard for representing free form curves and surfaces in computer aided design. Modelling is facilitated by control points whose positions determine the desired shape. NURBS possess many features that make them attractive for various applications such as geometric modelling, analysis, and approximation. However, NURBS suffer from a major drawback: control points need to form a rectangular topological grid.

We are interested in using NURBS for designing models in three-dimensional space. Consider the situation when a fine detail needs to be added to an existing coarse model. This is a typical operation performed in practice, for example, when adding a small ear detail to a face model. The structure of NURBS does not allow this to be performed as a local operation. If a new control point needs to be introduced, a whole strip of control points, running across the whole patch, has to be added. Otherwise, control points would no longer lie in a rectangular grid. Thus, requesting only a single new control point causes many unwanted control points to be introduced into the model. This fact complicates design and produces unnecessary overhead for the designer.

One of the earliest studies addressing this shortcoming led to the framework called hierarchical B-splines (HB-splines) [1]. Using nested spaces, the framework allows for locally refined patches that can represent finer detail. Later, a basis for these nested spaces was found and its stability studied [3]. More recently, HB-splines were studied from the point of view of iso-geometric analysis [4], a finite element framework [5]. By construction, the new basis functions formed by coarse and fine level B-splines do not sum to unity: weights need to be introduced. An improved construction, truncated hierarchical B-splines, which avoids the need for weights and provides a strongly stable basis, was recently discovered [6]. The truncated basis functions are convex combinations of B-splines.

Independently, spline spaces over T-meshes have been investigated [7,8]. In this approach to local refinement of B-splines, a T-mesh in the parameter space forms a foundation for the method. The most recent construction that addresses local refinement was coined locally refined (LR) B-splines [9].

The most widely known and used T-construction is T-splines [10,11]. T-splines are basically B-splines whose control meshes allow T-junctions. Local refinement is supported. Nevertheless, some

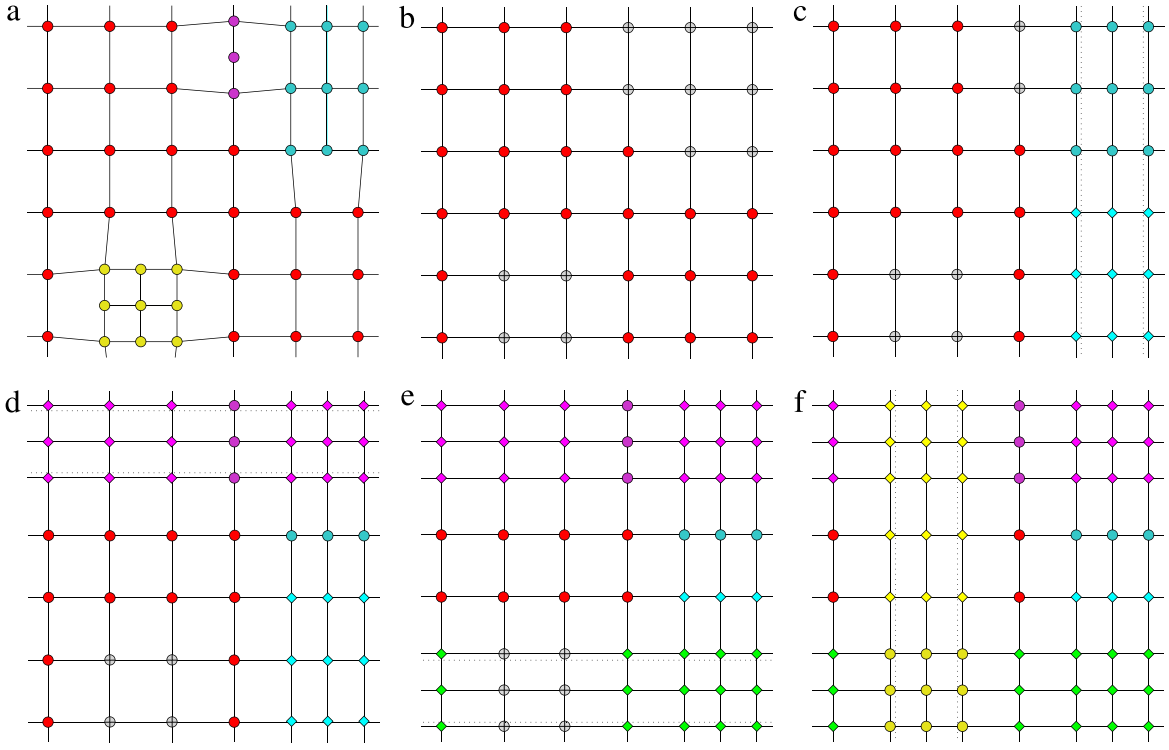

Fig. 2. An example of the basic idea in the cubic case. (a) The control mesh as seen by the user. In this example, the original mesh comprised the red control points. The user then requested a row of new points (blue, upper right), a single new point on a single edge (purple, middle top), and a square of new points (yellow, lower left). This is implemented as a series of layers. (b) The top layer is the original mesh. The grey points are those replaced by *visible* points in lower layers. (c) Inserting a row of new points requires a single new vertical knot. Diamonds show control points that are calculated from points in the layer above and circles show points that are revealed to the user for further manipulation. (d) Inserting a single control point still requires the insertion of an entire knot line. (e) A square of control points requires inserting two new knots. The first is inserted horizontally. (f) The second knot is inserted vertically. (f) shows the final tensor-product set of control points. The user-interface, (a), comprises the *visible* control points drawn from all the layers (coloured circles). Those control points in the higher layers that are not used are marked as *replaced* (grey circles) and are not manipulable and not used in any further calculations. Diamonds show *hidden* points calculated from the layer above. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)