Interactive Surface Design and Manipulation using PDE-Method through Autodesk Maya Plug-in

Eyad Elyan
School of Computing,
Robert Gordon University
Aberdeen, AB251HG, UK
Email: e.elyan@rgu.ac.uk

Hassan Ugail
School of Informatics
University of Bradford
Bradford, BD71DP, UK
h.ugail@brad.ac.uk

***Abstract*—This paper aims to propose a method for geometric design, modelling and shape manipulation using minimum input design parameters. Here, we address the method for the construction of 3D geometry based on the use of Elliptic Partial Differential Equations (PDE). The geometry corresponding to an object is treated as a set of surface patches, whereby each surface patch is represented using four boundary curves in the 3D space that formulate the appropriate boundary conditions for the chosen PDE. We present our methodology using a plug-in that was developed utilizing Maya API. The plug-in provides the user with tools that could be used easily and effectively for designing purposes. Maya is a popular 3D modelling tool. Various types of shapes with different complexities are presented here. Our proposed method allow the designer to utilize the Maya functionality for sketching curves in the 3D space that represents the outline of arbitrary shapes, construct the corresponding model using the PDE method, deform and sculpt these models interactively by editing the boundary curves.**

***Keywords*-PDE method; Geometric design; 3D modelling**

I. Introduction

The two dominant 3D geometry modelling tools in geometric modelling for representing graphic objects including complex objects such as aircrafts, human faces are polygonal based approaches and parametric patches. Polygonal representation is simple and flexible but the amount of data required is usually larger than those methods based on parametric representation. In addition, for interactive design polygonal approaches are not a suitable choice taking into consideration that designing an object within an interactive environment requires techniques which enable the user to edit and manipulate these shapes intuitively. Another crucial requirement in designing an object interactively is to keep the number of input design parameters to minimum. Moreover, when creating such tool for modelling and designing, implementation techniques have to be carefully designed to ensure efficiency which allows users to perform tasks on real time.

Parametric patch based techniques are dominantly based upon the spline techniques. Of these, B-Splines or more commonly referred as Non-Uniform Rational B-splines (NURBS) [1], [2] are considered as one of the dominant surface fitting techniques. The main reason why NURBS are widely used is because it offers smooth, compact parametric representation of data and at the same time it is widely supported by many commercial Computer Aided Design systems. Spline-based techniques were used extensively in interactive design. For example, parametric B spline patches have been used for modeling and animating a human face [3]. Similarly Song et. al [4] has demonstrated the use of B-Spline surface patches to construct parametric representation of a human face using 3D point cloud data. Huang and Yan [5] used NURBS curves and associated it with vertices of the facial data for modeling and animating 3D human face. Rappoport et al. [6] proposed a technique for interactive design of smooth objects using points displacement constraints, where the user defines an arbitrary number of control points and the system computes the object geometry subject to constraints satisfaction.

Although the spline-based techniques appeared to be promising from a computational point of view, they lack the intuitive feeling for direct manipulation of surfaces [7]. This is especially true if the user or the designer is not familiar with the mathematics on which the chosen geometric constraints are based.

In this paper we utilize the PDE-method for surface design, whereby a surface is generated by finding a solution to suitably chosen elliptic fourth order partial differential equations that satisfy certain boundary conditions [8]. PDE based modeling techniques has many advantages such as the capacity in using single surface patch to define a complex surfaces (e.g. human face), more powerful and flexible manipulation in shape control of surfaces. Furthermore, unlike spline-based techniques, it is more intuitive as the designers or users do not have to have a mathematical knowledge in order to manipulate the shape, taking into consideration that the boundary conditions (curves) are explicitly represented in the solution. Hence, surfaces of practical significance can be generated using a small se of design parameters.

Previous work using the PDE method has shown how generating complex-geometric shapes based on PDE method appeared to be powerful technique; examples include complex objects such as marine propeller [9], ship hulls [10] and aircraft [11]. In addition, graphical objects construction and efficient parameterization with relatively small number of design parameters was demonstrated using the PDE method in various publications [7], [12], [13], [14], [15]. For example Ugail et. al. [7] utilized the PDE method to demonstrate the creation and manipulation of complex geometries interactively in real time. Similarly, they demonstrated efficient parameterization and intuitive manipulation of complex shapes using PDE method [13] by editing and manipulating the boundary conditions.

The PDE method is used to generate graphical objects based on solving suitably chosen partial differential equations through a set of boundary conditions. In geometric design, it is common practice to define curves and surfaces using parametric representation. In simple words, if we consider $\underline{X}(u, v)$ being the definition of the surface in the Euclidean 3-space in a finite domain Ω , with a boundary $\partial\Omega$, on which boundary data is specified. Here we view the u and v to be the mapping from the point in Ω , and $\underline{X}(u, v)$ as a mapping from that point in WΩ to a point in the 3-space such that $R^2(\Omega) \to E^3$. To satisfy these requirements $\underline{X}$ is regarded to be the solution of a partial differential equation of the form

$$D^m_{uv}(\underline{X}) = \underline{F}(u, v) \quad (1)$$

where D^m_{uv} is a partial differential operator of order m in independent variable u and v, and F is a vector valued function of u and v . Thus the PDE method provides the user with tools by which she or he is able to specify regions in terms of boundary conditions and then the chosen PDE provides a mechanism to smooth the boundary data over the u and v parameter space, resulting in a smooth surface which is infinitely differentiable.

In this paper we aim to extend the existing work and show how surfaces with any arbitrary number of boundary conditions can be interactively constructed and manipulated in real time. In addition, we implement our solution as a plug-in to work within the Maya environment. This technique would allow users to benefit from the existing Maya functionalities to sketch and define 3D curves and utilize the PDE method to construct, sculpt and deform the corresponding graphical objects. It also provide a solid testing platform where real objects could be compared against PDE ones. Moreover, with this tool it modeling objects would be faster and more efficient in terms of computation and storage needs.

The reset of this paper is organized as follows. In the following section we briefly discuss the PDE method and provide some illustrative example. The following section w ill address the implementation of the solution within the MAYA environment. Extensive use of examples and illustrations will be provided and these examples have been created and manipulated on real time within MAYA environment. Finally conclusions will be drawn upon the discussion and future work will be suggested for further improvement.

II. PDE Method

The PDE method regards the generation of the parametric surface patch

$$\underline{X}(u, v) = (x(u, v), y(u, v), z(u, v) \quad (2)$$

as a solution to an elliptic partial differential equation. In particular, the PDE chosen for the work described in this paper is

$$\left(\frac{\partial^2}{\partial u^2} + \frac{\partial^2}{\partial v^2}\right)^2 \underline{X}(u, v) = 0 \quad (3)$$

In order to solve Equation (3) four boundary conditions are required. Here, we take these boundary conditions in the form of curves in the 3-space. These curves are taken to be the positions containing the data from the original objects whose geometry we wish to represent.

There is wide variety of methods for finding the solutions of the elliptic PDE in Equation (3). These include elementary separation of variables, Greens functions and numerical techniques. For the work described in this paper, where graphical objects need to be created, edited and recreated at real time, therefore we utilized an analytical solution scheme. We assume periodicity in the solution such that Ω is taken to be a finite domain defined as $\{\Omega : 0 \leq u \leq 1, 0 \leq v \leq 2\pi\}$ such that

$$\underline{X}(0, v) = \underline{P}_0(v) \quad (4)$$

$$\underline{X}(1, v) = \underline{P}_1(v) \quad (5)$$

$$\underline{X}_u(0, v) = \underline{d}_0(v) \quad (6)$$

$$\underline{X}_v(1, v) = \underline{P}_1(v) \quad (7)$$

where the boundary conditions $\underline{P}_0(v)$ and $\underline{P}_1(v)$ define the edges of the surface patch at $u = 0$ and $u = 1$ respectively. Based on this formulation and using the method of separation of variables, the analytic solution of Equation (3) can be written as

$$\underline{X}(u, v) = \underline{A}_0(u) + \sum_{n=1}^{\infty}[\underline{A}_n(u)cos(nv) + \underline{B}_n(u)sin(nv)] \quad (8)$$

where

$$\underline{A}_0 = \underline{a}_{00} + \underline{a}_{01}u + \underline{a}_{02}u^2 + \underline{a}_{03}u^3 \quad (9)$$

$$\underline{A}_n = \underline{a}_{n1}e^{anu} + \underline{a}_{n2}e^{anu} + \underline{a}_{n3}e^{-anu} + \underline{a}_{n4}e^{anu} \quad (10)$$

$$\underline{B}_n = \underline{b}_{n1}e^{anu} + \underline{b}_{n2}e^{anu} + \underline{b}_{n3}e^{-anu} + \underline{b}_{n4}e^{anu} \quad (11)$$

where $\underline{a}_{n1}, \underline{a}_{n2}, \underline{a}_{n3}, \underline{b}_{n1}, \underline{b}_{n2}, \underline{b}_{n3}$ and $\underline{b}_{n4}$ are vector-valued constants, whose values are determined by the imposed boundary conditions at $u = 0$ and $u = 1$. Often, Fourier analysis is performed in order to define the various constants for a given set of boundary conditions to identify the various Fourier coefficients. When the boundary conditions can be expressed exactly in terms of finite Fourier series, the solution given in Equation (8) is also finite. However, this is often not possible, in which case the solution will be the infinite series given in Equation (8). In this case an approximation technique is often deployed based (more details about the method could be found in [16]) on the sum of the first few Fourier modes and a remainder term i.e.

$$\underline{X}(u,v) = \underline{A}_0(u) + \sum_{n=1}^{\infty}[\underline{A}_n(u)cos(nv) + \underline{B}_n(u)sin(nv)] + \underline{R}(u,v) \tag{12}$$

Where $N \leq 6$ and $R(u,v)$ is a remainder function defined as

$$\underline{R}(u,v) = \underline{r}_1(v)e^{wu} + \underline{r}_2(v)e^{wu} + \underline{r}_3(v)e^{-wu} + \underline{r}_4(v)e^{-wu} \tag{13}$$

where $\underline{r}_1, \underline{r}_2, \underline{r}_3, \underline{r}_4$ and w are obtained by considering the difference between the original boundary conditions and the boundary conditions satisfied by the function

$$\underline{F}(u,v) = \underline{A}_0(u) + \sum_{n=1}^{N}[\underline{A}_n(u)cos(nv) + \underline{B}_n(u)sin(nv)] \tag{14}$$

The above solution technique is considerably faster than looking for a very accurate solution to Equation (3) using numerical methods such as finite element or finite difference. It is important to emphasize here, that although the solution is approximate, it guarantees that the chosen boundary conditions are satisfied by the function $\underline{F}(u,v)$.

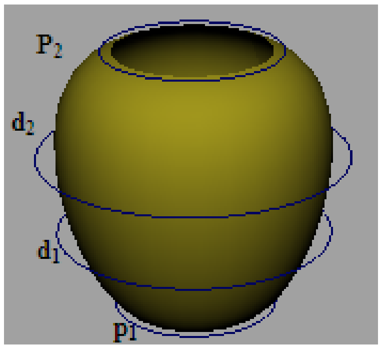


Figure 1. PDE surface generated subject to the boundary conditions shown in the image.

As an example consider the curves and the shape shown in Figure 1. Here the curves define the necessary boundary conditions for solving the Equation (3). It is important to point out that the analytic solution discussed above is applicable to periodic conditions and therefore, we restrict the domain of the solution to $\{\Omega : 0 \leq u \leq 1, 0 \leq v \leq 2\pi\}$ which adequately describes the boundary conditions shown in Figure 1. Although the boundary curves were deliberately scaled to be outside the surface patch for illustrative purposes in Figure 1 it should be noted that the resulting PDE surface patch shown in here actually passes through the boundary conditions $p1, d1, d2$ and $p2$ respectively whereby the PDE enables a smooth interpolation of these curves. It is also worth highlighting that the resulting PDE surface is solely controlled by the four boundary curves as will be further discussed in the following sections.

III. MAYA IMPLEMENTATION

From an implementation point of view, to construct one surface patch, the designer/ user has to use four different boundary curves. Nevertheless, when modeling complex objects, four curves hardly provide sufficient information to depict such objects precisely (i.e. human face, [17]). In such cases more than one surface patch is needed, each of which represented and controlled by a group of boundary curves (four curves/ surface patch). Hence, surface generation becomes blending of the different surface patches, each of which is generated by four consecutive boundary curves and each two adjacent surface patches share a common boundary curves at their edge to guarantee continuity and smoothness along the blended surface patches. This means that for instance to construct an object where five surface patches are required, then five variant fourth order PDEs have to be resolved. This means that efficiency and speed to solve PDE surfaces is a very critical issue.

In order to take full advantage of the powerful functions of MAYA [18], we implement the above mathematical formulation at two stages. On the first stage, a set of objects were designed and implemented to solve elliptic partial differential equations. These classes were implemented using C++, where memory is allocated when it is required dynamically and released once it is no more needed. This is of paramount importance when solving PDE's in real time, where resources utilization is optimized.

At the second stage of implementing the PDE solution we utilized MAYA 8.5 package through the API using C++. The most important issue when constructing PDE surface is the boundary conditions that defines the outline of the object to be designed. In this case these conditions are curves in the 3-Space. Here, MAYA tools for drawing 3-space curves are used to generate any number of curves that represent one or more surface patch to be constructed using the PDE method. Taking advantage of the MAYA Nodes, a node called PDENode was developed. The input to this node is simple a set of 3-space curves (minimum four curves required) and the output is a PDE surface. The general structure of the MAYA implementation is shown in Figure 2.

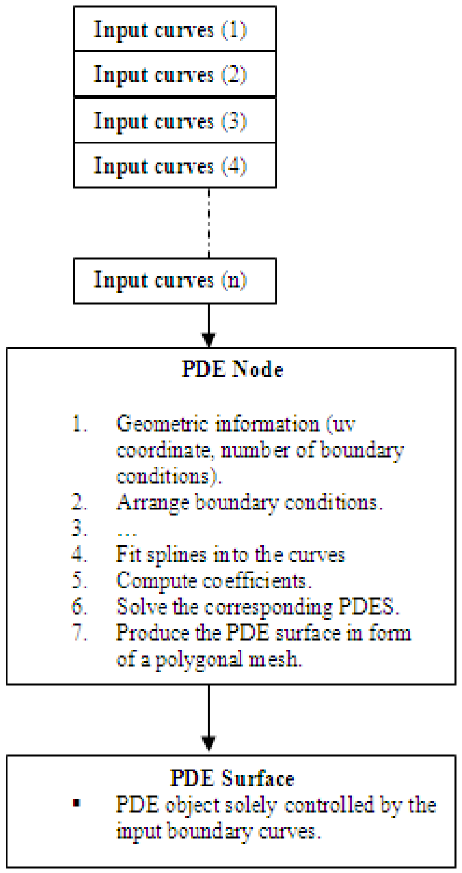

Input curves (1)
Input curves (2)
Input curves (3)
Input curves (4)
Input curves (n)
PDE Node
1. Geometric information (uv coordinate, number of boundary conditions).
2. Arrange boundary conditions.
3. ...
4. Fit splines into the curves
5. Compute coefficients.
6. Solve the corresponding PDES.
7. Produce the PDE surface in form of a polygonal mesh.
PDE Surface
▪ PDE object solely controlled by the input boundary curves.

Figure 2. General structure of the MAYA implementation to solve PDE surfaces.

In MAYA data are transferred between networks of nodes in the dependency graph. Every node has a set of input parameters and has an output. A node output could be simply an input to another. Figure 2 shows the implantation of the PDE node. Once the node receives an input which is a set of input curves, it performs the necessary computations and produces an output that is the PDE surface represented as a node within the dependency graph of a MAYA scene. In a practical scenario, the user would plot a set of curves that defines that outline of a particular object, then he/she will issue a command (magicCurves) resulting in contracting the PDE node with its output as a PDE surface. Clearly any changes to the boundary conditions will trigger the PDE node compute method to recompute the surface subject its adjusted boundary conditions. In addition, the resulting PDE surface could be edited using the attribute editor, where a set of parameters could be adjusted accordingly, such as the u, v coordinates, the finite domain of the PDE solution Ω and the number of control points on each boundary condition. Figure3 demonstrates the use of the plug-in using simple PDE surface composed of two surface patches, controlled by 7 boundary curves, whereby some of the surface attributes have been edited using the attribute editor.

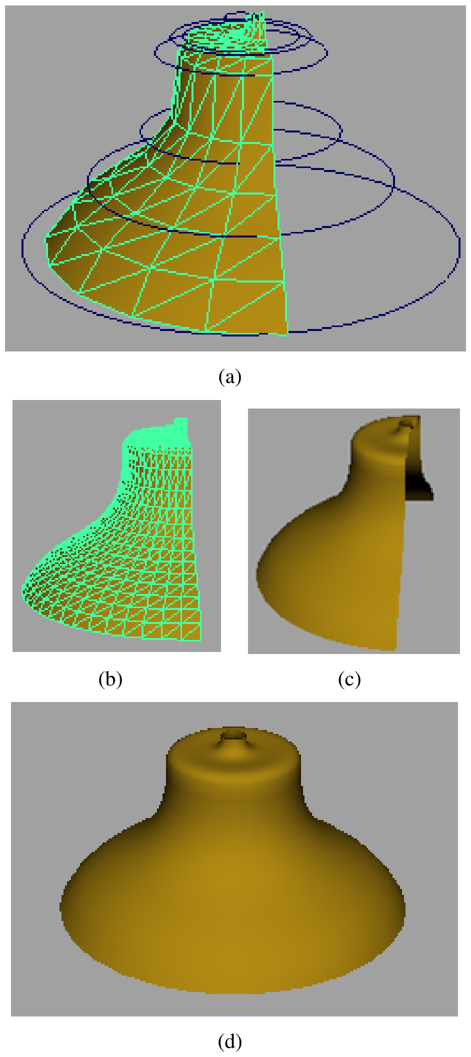


Figure 3. constructing and editing PDE objects interactively using MAYA plug-in and the attribute editor

Figure3 (a) shows the initial PDE surface controlled by 7 boundary curves and composed of two surface patches. Here, a low resolution mesh made of 10 x 10 has been produced. In Figure3 (b) using the attribute editor the resolution of the mesh of the PDE surface has been increased to be 25 x 30, which result a smoother surface as shown without the wireframe in Figure3 (c). It is also shown in Figure3(a), (b) and (c) that the domain of the solution was restricted to be between 0 and π while in Figure3 (d) the surface has been resolved after adjusting the domain to 2π.

It is worth pointing out that the domain of the solution could be adjusted to any arbitrary value between 0 and 2π as shown in Figure4.

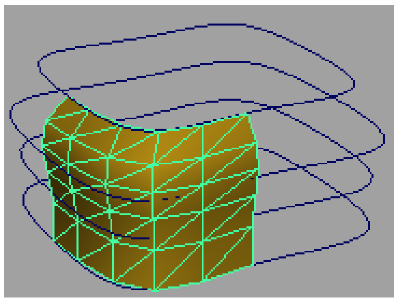


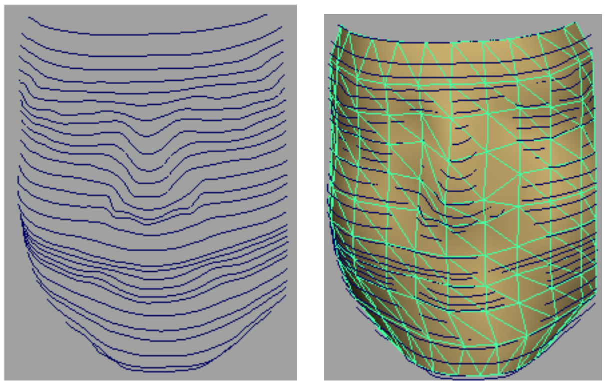
Figure 4. PDE surface restricted within the domain between 0 and $\frac{\pi}{2}$.

Editing the PDE surface properties interactively provide the user with a flexible tool to design complex shapes by just drawing the outlines of these shapes using.

IV. Complex Example

Here we consider the human face as a complex graphical object to test our prototype. There is several ways to segment the face into meaningful regions, whereby each region represents a certain feature of the face e.g. eyes region, nose, forehead, etc Here, we segment the face onto nine different patches; each of them is controlled by a set of boundary curves and represents certain characterizing feature of the human face. These patches represents the forehead area which is controlled by four boundary curves, the eyes area controlled by 7 curves (e.g. two surface patches), the nose area controlled by 4 curves, the mouth and the chin area, controlled by 7 and 4 curves respectively. In total, 28 cross-sectional curves represent the boundary conditions of the facial image.

In this example we used a 3D scanned image, extract a set of cross-sectional curves that represent the above mentioned areas and then generate the PDE surface as a surface composed of nine surface patches. Figure 5 (a) shows the arrangement of the boundary conditions of the facial data.



(a) (b)

Figure 5. PDE facial data representation, (a) boundary conditions, (b) initial PDE surface representing a human face.

The PDE surface shown in Figure 5 (b) is an initial PDE surface with low mesh resolution 10 x 10. Figure 6 shows an improved surface by increasing the mesh resolution (edit the uv dimensions via the plug-in attribute editor). It is important to note here, that any changes to any of the surface attribute would essentially trigger the PDE node to re-compute the PDE surface.

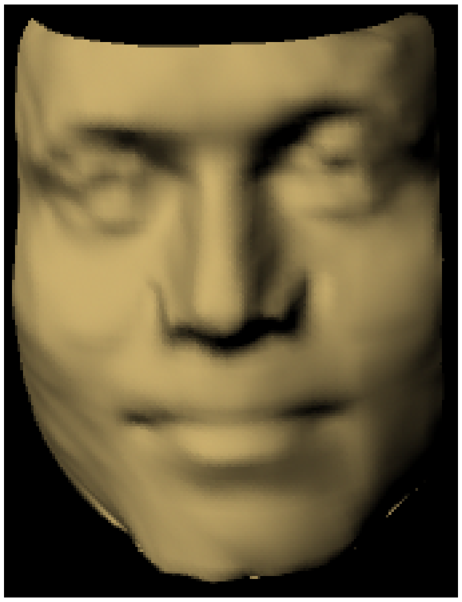
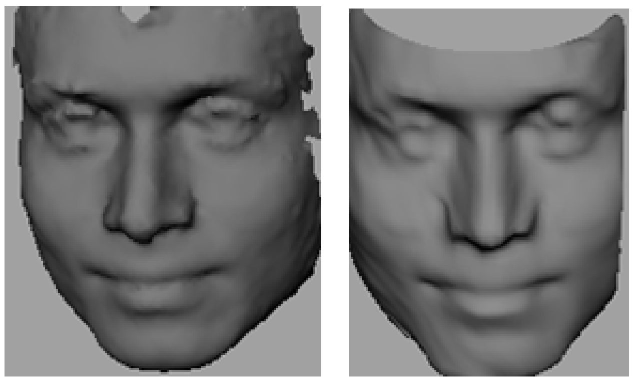


Figure 6. PDE facial surface with an increased mesh resolution, the surface is controlled by the same set of boundary curves shown in Figure 5 (a).

With the implementation of the mathematical formulation of the PDE method within the MAYA environment it becomes possible and easier to compare an original surface with its corresponding PDE as shown in Figure 7 and 8 respectively.



(a) (b)

Figure 7. comparing original 3D facial image, with a PDE image, (a) original 3D image, (b) corresponding PDE image.

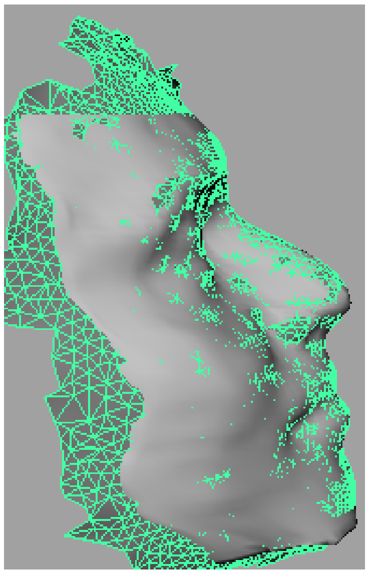


Figure 8. comparing original 3D facial image, with a PDE image, by putting them on top of each other.

V. Conclusion and Future Work

The prototype of the plug-in was tested on Pentium 4 machine, with 1 GB memory. All the editing and manipulation of the PDE graphical objects were conducted on real time. We have successfully utilized the PDE method to generate complex graphical entities. These entities are composed of one or more surface patch. In addition, we provided the user with a graphical and user-friendly tool within the MAYA environment to generate, edit and manipulate such complex objects on real time.

The manipulation of the PDE surfaces is carried out by editing the boundary curves using some of the existing MAYA features (e.g. move, rotate, scale). This requires editing the positional control points of the curves. As a possible improvement on the current prototype, we intend to make this task more intuitive by applying a local subdivision scheme. Here, the user will be allowed to allocate two sets of boundary curves. The first set is to control the PDE surface, just as explained on the previous sections, while the second set of curves will define a local PDE surface (that is hidden from the user) which will be used to edit the surface by establishing point correspondence between the local and global PDE surfaces.

Acknowledgment

The authors would like to thank the University of Bradford who funded this research.

References

[1] I. D. Faux and M. J. Pratt, *Computational Geometry for Design and Manufacture*. New York, NY, USA: Halsted Press, 1979.

[2] G. Farin, “From conics to nurbs: A tutorial and survey,” *IEEE Computer Graphics and Applications*, vol. 12, no. 5, pp. 78–86, 1992.

[3] M. Hoch, G. Fleischmann, and B. Girod, “Modeling and animation of facial expressions based on b-splines,” *The Visual Computer*, vol. 2, pp. 87–95, 1994.

[4] Y. Song, L. Bai, and Y. Wang, “3d object modelling for entertainment applications,” New York, NY, USA, p. 62, 2006.

[5] D. Huang and H. Yan, “Modeling and animation of human expressions using nurbs curves based on facial anatomy,” *Signal Processing: Image Communication*, vol. 17, no. 6, pp. 457 – 465, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/B6V08-45NGM5V-1/2/7b6e4488ecc369e9bf0e310f1e419d43

[6] A. Rappoport, Y. Hel-Or, and M. Werman, “Interactive design of smooth objects with probabilistic point constraints,” *ACM Trans. Graph.*, vol. 13, no. 2, pp. 156–176, 1994.

[7] H. Ugail, M. I. G. Bloor, and M. J. Wilson, “Techniques for interactive design using the pde method,” *ACM Trans. Graph.*, vol. 18, no. 2, pp. 195–212, 1999.

[8] M. I. G. Bloor and M. J. Wilson, “Generating blend surfaces using partial differential equations,” *Comput. Aided Des.*, vol. 21, no. 3, pp. 165–171, 1989.

[9] C. W. Dekanski, M. Bloor, and M. Wilson, “The representation of marine propeller blades using the pde method,” *Journal of Ship Research*, vol. 38, pp. 319–328, 1995.

[10] T. LOWE, M. Bloor, and M. Wilson, “The automatic design of hull surface geometries,” *Journal of Ship Research*, vol. 38, pp. 319–328, 1994.

[11] M. I. G. Bloor and M. J. Wilson, “Efficient parameterization of aircraft geometry,” *J Aircraft*, vol. 32, pp. 1269–1275, 1995.

[12] H. Du and H. Qin, “A shape design system using volumetric implicit pdes,” *Computer-Aided Design*, vol. 36, no. 11, pp. 1101 – 1116, 2004, solid Modeling Theory and Applications. [Online]. Available: http://www.sciencedirect.com/science/article/B6TYR-4BM8T22-1/2/b5b356042c619bc3685bf5297b36ec63

[13] H. Ugail, M. I. G. Bloor, and M. J. Wilson, “Manipulation of pde surfaces using an interactively defined parameterisation,” *Computers & Graphics*, vol. 23, no. 4, pp. 525–534, 1999.

[14] H. Du and H. Qin, “Direct manipulation and interactive sculpting of pde surfaces,” 2000.

[15] M. Bloor, M. Wilson, Z. Knudsen, C. Knudsen, and A. Holden, “Time-dependent parametric surface models of the human heart,” 1999, pp. 174–179, 246.

[16] M. I. G. Bloor and M. J. Wilson, “Spectral approximations to pde surfaces,” *J Computer-Aided Design*, vol. 29, pp. 145–152, 1995.

[17] E. Elyan and H. Ugail, “Reconstruction of 3d human facial images using partial differential equations,” *Journal of Computers (JCP)*, vol. 2, pp. 1–8, 2007.

[18] “http://www.autodesk.com/maya.”