GFFD: General Free Form Deformations using Partition Unity Parametrics

Ali Mahdavi-Amiri
Faramarz Samavati

April 8, 2017

Abstract

Free Form Deformations (FFD) have been successfully employed to deform 2D or 3D shapes to manipulate them and obtain a desired shape. In this deformation, a lattice of control points is placed on the given shape and by moving control points, the given shape is deformed according to a set of underlying smooth basis function (e.g. BSpline, NURBS). In this paper, we attempt to generalize Free Form Deformations (GFFD) by using more general basis functions called Partition Uniform Parametrics or PUPs. We provide a comparison of GFFD in which PUPs are employed as the basis functions with BSpline, Bezier, and NURBS basis functions. Although this work is in its preliminary stages, we believe that there are many directions to improve and extend the current idea. We eventually discuss these ideas in the paper.
keywords: FFD, PUPS, BSpline.

1 Introduction

One of most important and well studied subjects in Computer Graphics is deformation. Both curve deformations and surface deformations have been the subject of many research works. While different approaches have been taken for deforming a curve or surface that we will discuss in Related Work section, all these deformation techniques share some commonalities. A deformation technique should be easy to compute, intuitive for the user, and simple for interaction. Despite the existence of these commonalities, the approaches taken to handle curves or surface deformations can be quite different.

Not only the deformation approaches can be very different, but the application of deformations can also vary. There are many reasons that one wants to deform a mesh or curve including animation, physical simulations, or mesh and curve editing [10, 5, 7, 3]. The application of deformation is one of the reasons that different deformation techniques with different interaction possibilities have been created. For instance, one of the common techniques to deform a surface is to use a sketch-based environment in which users can provide a set of curves and the mesh is deformed according to the curve see (Figure 1).

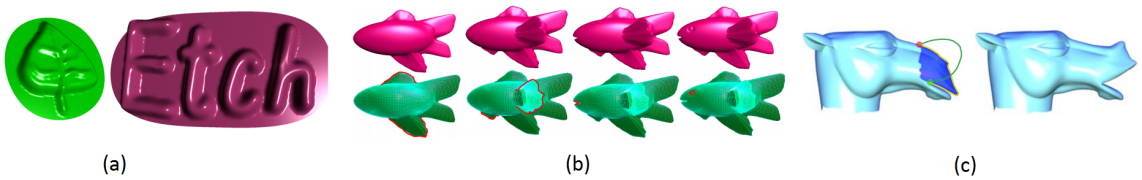

Figure 1: Sketch-based deformation of a given 3D mesh. Images are take from [21] (a), [14] (b), and [18] (c).

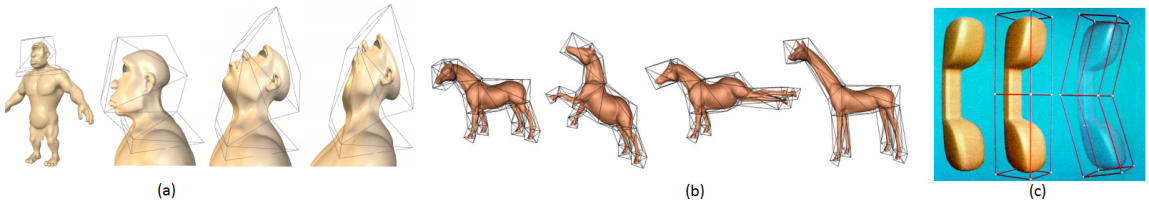

Figure 2: Cage-based deformation of a given 3D mesh. Images are take from [13] (a), [12] (b), and [24] (c).

In addition to sketch-based deformation, one of the most useful and easy to use deformation techniques is *space deformation* in which a shape like a curve or a surface is embedded in a 2D or 3D lattice of control points called *cage* and the shape is deformed when the control points of this lattice is moved (see Figure 2). The movement of the lattice control points impacts the vertices of the shape based on a set of *basis functions*. These basis functions usually are chosen as smooth basis functions that provide particular properties such as convex-hull or affinity. Such properties provide a more intuitive environment for the user and the result of the deformation is predictable.

Different types of basis functions have been used for a cage-based deformation such as Bezier, BSpline, and NURBS. These basis functions have been the industry standard for animation, modeling, and computer aided design for a long period of time. In this paper, we want to study the possibility of replacing these basis functions with similar yet more general basis functions called Partition Unity Parametrics (PUPs). PUPs have been successfully employed in meta modeling, and subdivision, and texture generation [23, 17, 4]. However, providing an intuitive deformation technique using these powerful basis functions are still under question. In this paper, we provide their behaviors for some simple curve cases and raise some interesting questions that are needed to be answered in order to successfully employ PUPs for cage-based deformations.

2 Related Work

When a high resolution curve or mesh is supposed to be deformed, one possibility is to change the position of every single vertex which is a tedious and almost impossible task for a real application. As a result, deformations are usually defined in such a was that a group of vertices (usually in the same neighborhood) are deformed at once. We can loosely divide techniques performing these tasks into two main categories: surface/curve based deformation and handle based deformation. In the following we discuss each category and the techniques that have been proposed for them.

2.1 Surface/curve based deformation

In this approach of the deformation, information about the curve or surface (e.g, connectivity of vertices) is extensively used to perform the deformation. Since manipulating every single vertex is a tedious task, an approach is needed to select a group of vertices and modify their locations. One way to do such a task is through multiresolution mesh editing. In this approach, high resolution mesh is initially decomposed to a low resolution mesh. Then few vertices are selected and manipulated. The high resolution mesh is then re-created by an upsampling technique such as a smooth subdivision [20, 19, 27, 1].

Another method to select a group of vertices and propagate the deformation through vertices is to use Laplacian mesh editing technique. In this approach, a vertex is modified and then neighbors around this vertex are adjusted through an energy minimization by minimizing the Laplacian of vertices affected by the deformation [25].

While vertex manipulation for deforming a mesh has been extensively used, sketch-based deformation paradigm is also very useful [26]. In this paradigm, users provide a curve introducing the shape and magnitude of deformation. The mesh is then deformed respecting the curve and satisfying some smooth conditions such as smooth subdivision filters [21, 14], or Laplacian [18].

2.2 Handle based deformation

In this type of deformation, vertices of the mesh or curve are not directly manipulated and some extra handles are provided for the users to facilitate the deformation. There are three types of handles that are usually provided for the users: a point, skeleton or cage. In point-based deformation, some unconnected points are provided for the use. Users change the location of these points and the mesh or curve is deformed with respect to these points satisfying some constraints mainly smoothness [9]. In skeleton-based deformation, a connected graph representing the skeleton of the object is provided and when a node of this graph is relocated, the portion of the object corresponding to that moved piece of skeleton are deformed (see Figure 3).

In this paper, we are mostly interested in cage-based deformation. Cage-based deformation provides a control net embedding the shape that is supposed to get deformed.

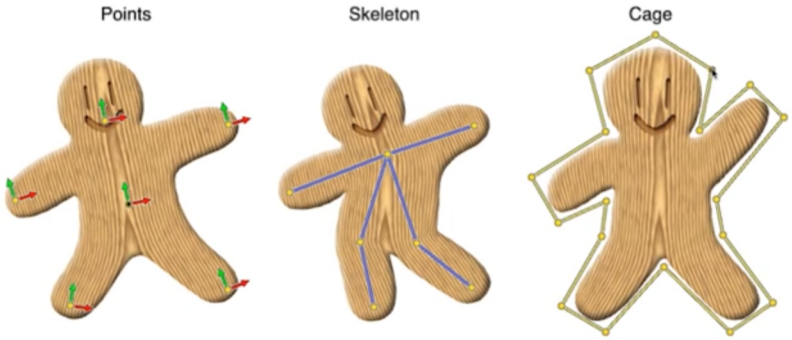

Figure 3: Point-based, skeleton-based, and cage-based deformations [9].

Using the control net, a coordinate system is defined for the points of the shape that can be smoothly deformed when the control net is manipulated. There are many possibilities for choosing the such coordinate system such as harmonic, mean-value and green coordinate systems [11, 6, 12, 13].

While there are versions of the cage-based deformation in which the cage can be concave and irregular [12, 11], we are interested in a variation of cage-based deformations called Free Form Deformations or FFD in which the control net is a lattice of points regularly connected to each other [24, 8]. In FFD, when a control net is manipulated, this manipulation is propagated to the shape through a set of basis functions such as BSpline, Bezier, or NURBS [24, 8]. However, in this paper, we are interested in the study of replacing basis functions of FFD with a general form of NURBS that is called Partition Unity of Parametrics (PUPs). In Section 3, we first provide a more detailed overview of FFD and then discuss PUPs in Section 4.

3 Overview of FFD

In FFD, a regular lattice of points are initially placed on the shape that is supposed to be deformed (see Figure 4 (Left)). Now, it is expected when a point of this lattice is moved, the shape is deformed accordingly (see Figure 4 (Right)). As a result, there should be a meaningful connection between the control points and vertices of the shape. One of the most useful and simple forms of making such a relationship is to use the lattice as the parameter domain of a Bezier patch and points of the lattice play the role of the control points of the Bezier patch. If point, X, has (s,t) coordinate on the parameter domain of the Bezier patch, its location is found as $X(s,t) = \sum_{j=0}^{n}\sum_{i=0}^{m} B_i^m(s)B_j^n(t)P_{ij}$, where B_i^m is ith Bernstein polynomial of degree m.

Finding s and t is an important task to set up the FFD. In Bezier case, it is easy to find s and t. If the bottom-left point of the lattice, $X_{BL} = (x_{min}, y_{min})$, and the top-right

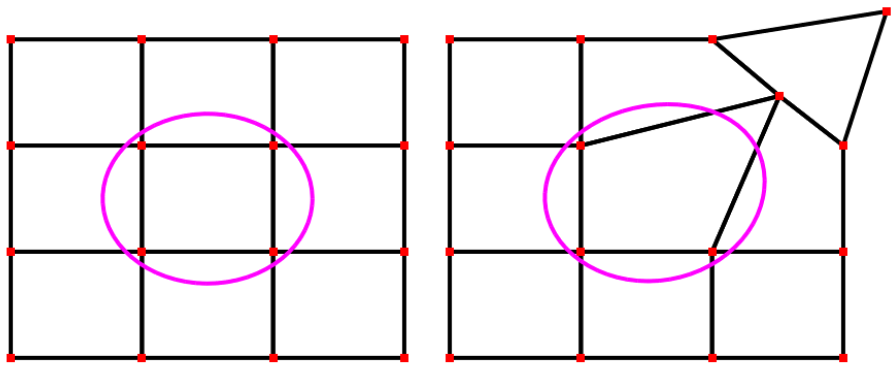

Figure 4: A shape (circle) is embedded in a lattice of points (Left). When points of the lattice are modified, the shape is deformed accordingly.

point, $X_{TR} = (x_{max}, y_{max})$ are given, then $s = \frac{x - x_{min}}{x_{max} - x_{min}}$ and $t = \frac{y - y_{min}}{y_{max} - y_{min}}$ for point $p = (x, y)$ (see Figure 5(Left)).

To get a uniform behavior of the control points in FFD, they need to be placed uniformly along x and y axes. We can easily place the points uniformly having X_{BL} and X_{TR} as:

$$P_{(i,j)} = (x_{min} + \frac{i}{m}(x_{max} - x_{min}), y_{min} + \frac{j}{n}(y_{max} - x_{min})).$$

Bernstein polynomial is not the only option for the basis function of FFD. It is possible to use other types of basis functions such as BSpline or Bezier. The process of finding (s, t) is similar but there are some advantages for using BSpline over NURBS. Using Bernstein polynomials, when the number of points increases to have a larger number of control points for manipulation, the order of the basis function also increases. This has two artifacts, one is that it makes the calculations more complex and inefficient. Second is that the deformations become smoother due to the smoothness of higher order basis functions. As a result, sharper modifications are more difficult having a larger number of control points in FFD with Bernstein polynomial basis functions. BSplines, however do not have such a problem, as one can use a low degree BSpline basis function for a control net with a large number of control points [2]. For instance, Figure 6(a) illustrates a second order BSpline on a 4×4 lattice.

Although BSplines are more general and applicable than Bernstein polynomials, they are limited to only uniform deformations. This means that all of the control points have exactly the same effect on deforming the shape. NURBS are generalizations of BSpline basis functions in which each control point, $P_{i,j}$, can have a weight, $w_{i,j}$. As a result, the effect of a particular point on the shape can be controlled (see Figure 6) (c). Since multiplying the points with different weights might violate sum to unity property of basis

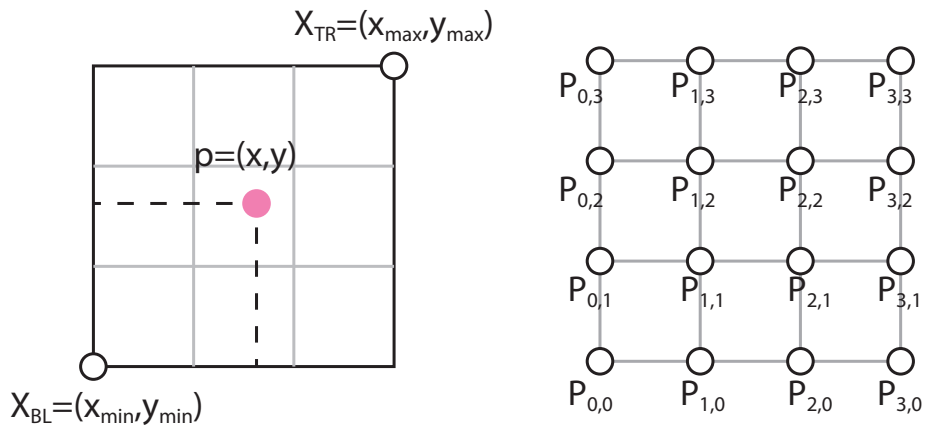

Figure 5: Left: parameters of a given point, P are calculated using bottom-left point X_{BL} and top-right point X_{TR}. Right: Control points of the lattice of FFD.

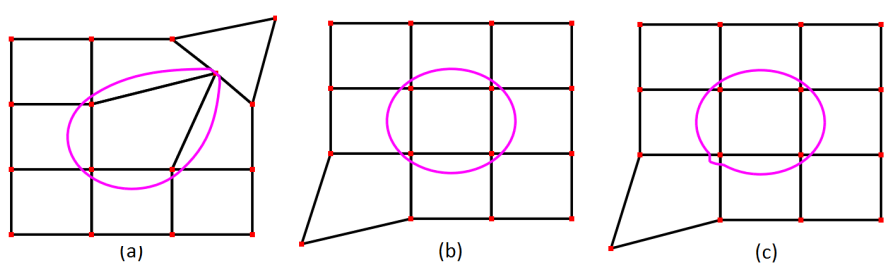

Figure 6: (a) A second order BSpline basis function on a 4×4 lattice. (b) A second order NURBS with uniform weights (BSpline) (c) A second order NURBS with weight 10 for $p_{0,0}$ and one for others.

functions, a normalization is used to preserve the affinity as:

$$X(s,t) = \frac{\sum_{j=0}^{n}\sum_{i=0}^{m} w_{i,j} P_{ij} N_i^m(s) N_j^n(t)}{\sum_{j=0}^{n}\sum_{i=0}^{m} w_{i,j} N_i^m(s) N_j^n(t)}.$$

in which $N_i^m(s)$ is ith BSpline basis function with order m.

Both NURBS and BSplines have been extensively employed as industrial standards for deformations. However, both of these basis functions are limited to particular pre-defined basis functions. For instance, it is not possible to have a high-degree (very smooth) basis functions for a lattice with a low number of control points as the degree of BSpline and NURBS increases when the number of control points increases. Moreover, other interesting basis functions such as arbitrary non-smooth functions are not possible and we cannot have different basis functions for different control points. PUPs, however, do not have any of these limitations and they can be used in a much more general setting for deforming object in 3D and 2D. Although, we do not study all the possibilities that PUPs provide for deformations, we provide a discussion of the possibilities that PUPs offer in this paper.

4 Overview of PUPs

Partition Unity Parametrics or PUPs are generalization of NURBS. As discussed earlier, NURBS patches are defined as:

$$X(s,t) = \frac{\sum_{j=0}^{n}\sum_{i=0}^{m} w_{i,j} P_{ij} N_i^m(s) N_j^n(t)}{\sum_{j=0}^{n}\sum_{i=0}^{m} w_{i,j} N_i^m(s) N_j^n(t)}$$

where $N_i^m(s)$ is ith BSpline basis function with order m. In PUPs, the BSpline basis functions are replaced by arbitrary weight functions [23]. In fact, we can define basis functions of PUPs as:

$$R_{i,j}(s,t) = \frac{W_{i,j}(s,t)}{\sum_{j=0}^{n} W_j(s,t)}.$$

This way, sum to unity property is guaranteed while $R_{i,j}(s,t)$ can be defined freely by modifying $W_{i,j}(s,t)$. As discussed in [23], one of the options for $W_{i,j}(s,t)$ can be defined as:

$$W_{i,j}(s,t) = w_{i,j} N_{i,m}(\frac{s-i}{c_i}) N_{j,n}(\frac{t-j}{c_j}).$$

Note that we have used a two dimensional explanation of PUPS for 2D FFD. One dimensional and three dimensional versions of this definition result a curve and a 3D FFD respectively.

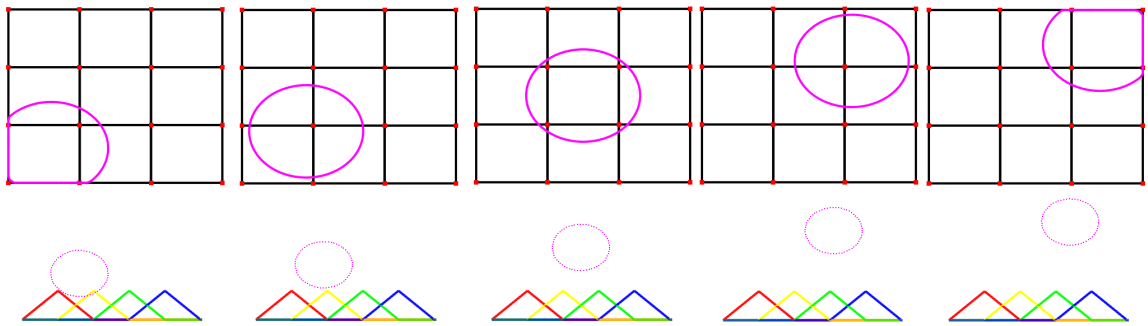

Figure 7: Moving (s, t) in the parameter domain and its effect on the shape.

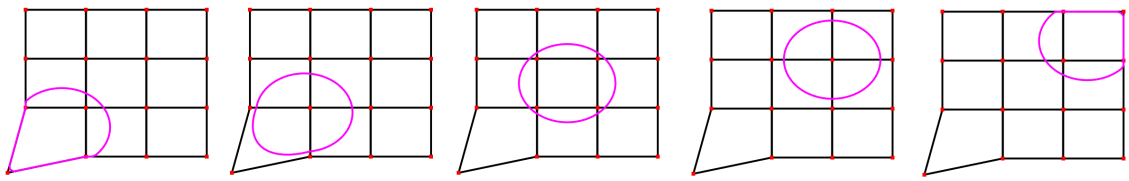

Figure 8: Deforming the shape with the same parameter and basis functions as Figure 7.

5 GFFD using PUPs

The structure of lattices and basis functions of GFFD with PUPs are fairly similar to FFD with Bezier or NURBS. The difference is that we have the freedom to use different basis functions for each control point with different span, order, and location on the parameter domain. Basis functions can be even non-polynomial. The only constraint that we should enforce to the system is to normalize the basis functions over the parameter domain to obtain an affine invariant deformation.

This freedom may also provide some artifacts. For instance freely modifying the basis functions over the parameter domain may result to have asymmetric deformations that might not be intuitive. In this section, we describe the effect of choosing basis functions and moving parameters over parameter domain.

Note that parameters (s, t), can be defined anywhere on the parameter domain as long as at least one basis function exist on s and t. For the first example, we use regular second order basis functions for both vertical and horizontal axes. As apparent in Figure 7, we have a symmetric effect on the symmetric circle as expected. This shows that although we are using the same basis functions as BSpline, we can have different effects on the deformation by changing the parameters while we can get BSpline too as shown in the middle case for both Figures 7 and 8.

In addition to translating the parameters, we can also scale the parameters and obtain different results. Scaling parameters up results to have more basis functions involved in

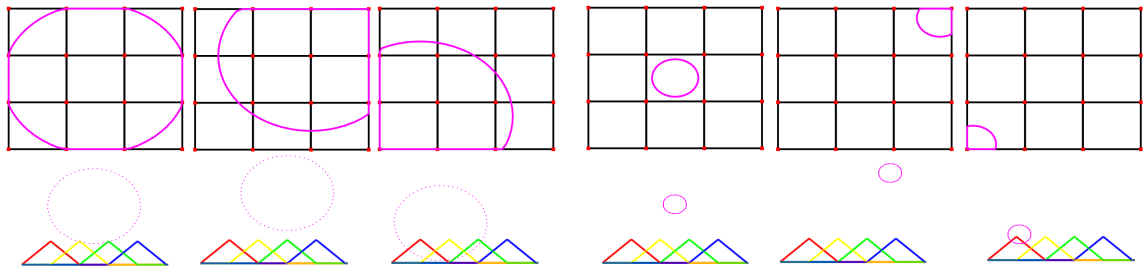

Figure 9: Up-scaling and down-scaling of the parameters in parameter domain.

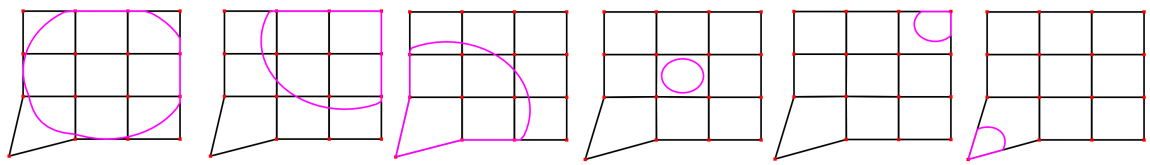

Figure 10: Deforming the shape with the same parameter and basis functions as Figure 9.

the final deformed shape while scaling down the parameters results less basis functions involved. Figure 9 illustrates the effects of up-scaling and down-scaling of the parameters in parameter domain. As illustrated in Figure 14, the effect of modifying control points has been changed by changing the scale of parameters while the linearity of the basis function is still visible in the deformation.

Another parameter in changing GFFD deformation with PUPs basis functions that use BSpline is the parameter c_i (see Section 4). By changing c_i, we can get different deformation results as the span of basis functions are changed and basis functions may have more overlaps. Figure 11 illustrates the effect of c_i on the shape of the curve. In this example $c_i = 2$ and basis functions are linear. Figure 12 shows the effect of deformations using these basis functions. Figures 13 and 14 illustrate the same span ($c_i = 2$) for third order and fourth order basis functions

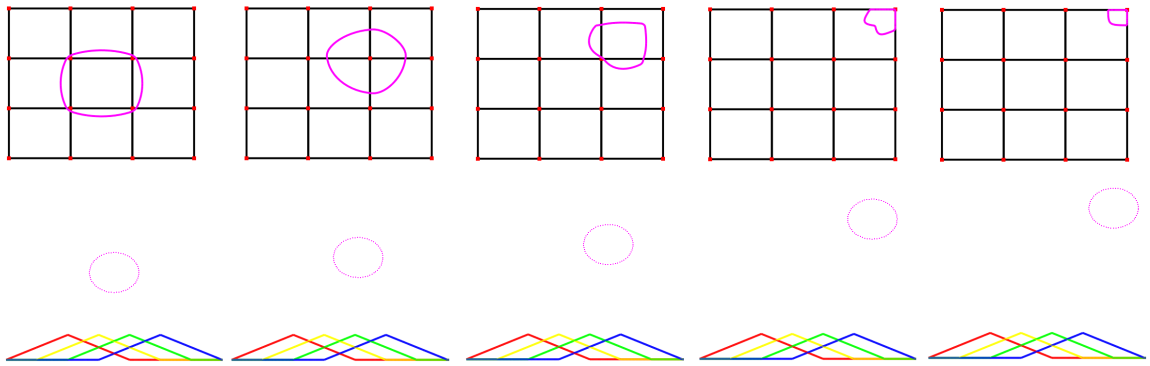

Figure 11: Second order BSpline basis functions with $c_i = 2$.

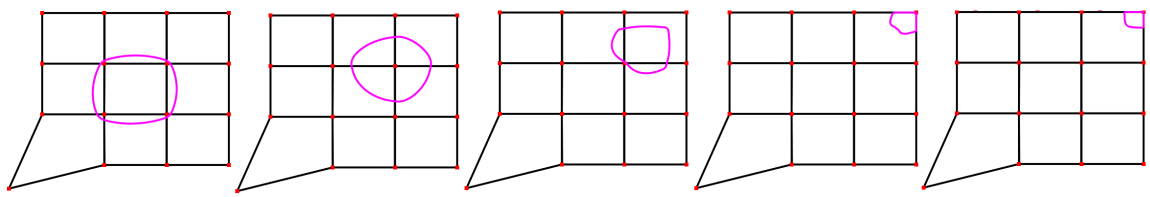

Figure 12: Deformation for the parameter domain and basis functions in Figure 9.

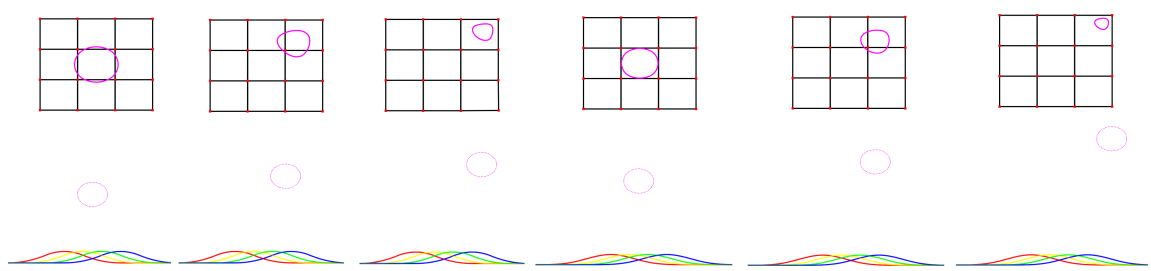

Figure 13: Third and fourth order BSpline basis functions with $c_i = 2$. The first three are third order.

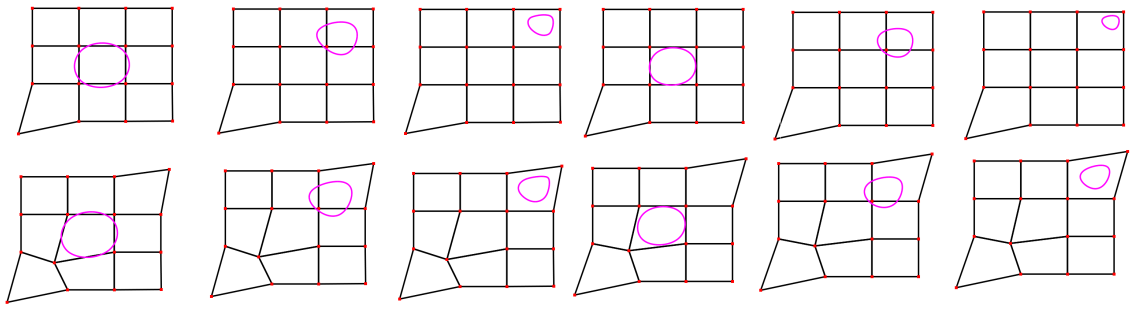

Figure 14: Deformation for the parameter domain and basis functions of Figure 13.

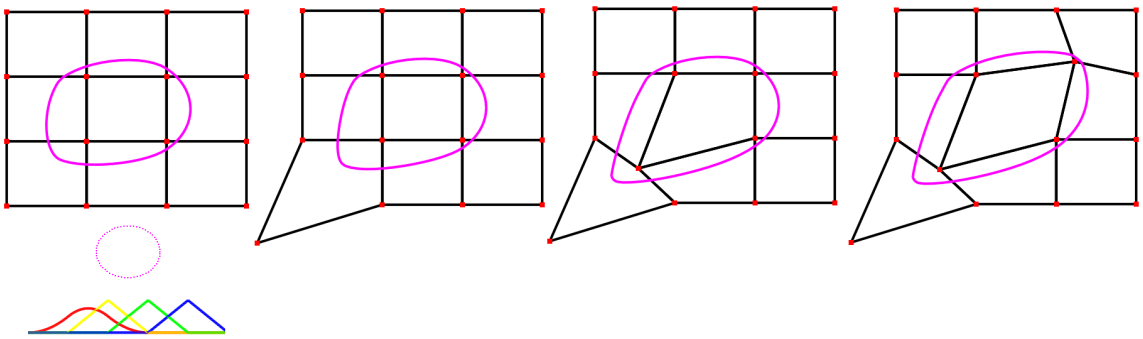

Figure 15: Having a third order BSpline basis function as the first basis function along x and y axes while all the other basis functions are linear.

An interesting fact about basis functions of pups is that we can combine in any fashion that we like. Another possibility is to mix and match basis functions with different orders and spans in order to emphasize on a particular control point or give it any specific style. Figure 15 illustrates such scenario when the first basis function along x and y axes are third order and all the other basis functions are of second order.

In this Section, we discussed the parameters of PUPs on which we have freedom and possibilities to manipulate in FFD, We also illustrated the effects of changing these parameters on the final deformed object. In next Section, we discuss some future works that can be done in order to get a powerful deformation framework using PUPs.

6 Future Work and Conclusion

In the original work of PUPs, to apply a deformation to a mesh, a parameterization of the mesh is created first and then control points and basis functions are added to the shape. Using GFFD, deforming a mesh is fairly straightforward by using a three-dimensional cage

instead of a 2D dimensional cage and convolution of three PUPs basis functions together. An effortless future work is to implement a 3D cage for meshes using PUPs basis functions. Another future direction is to use these basis functions on a general form lattice that can be represented by ACM [15, 16]. This task is more challenging as mixing basis functions at singular points is not straightforward. The most important future work is to provide a set-up and interactive environment for users to obtain a meaningful deformations out of PUPs. Although all the nuts and bolts of a GFFS is ready, combining all these in a user friendly environment is a future work. In this paper, we only used BSpline basis functions for the underlying PUP's basis function structure, using other types of basis functions such as CINPACT [22] and studying their behaviors in GFFD environment can be an interesting future research. In addition to these future research, a good question is that whether we can provide a point-based or skeleton-based deforamtions using PUPs. In [9], some criteria have been proposed for a good basis function which PUPs already attain some of those (e.g. sum to unity). The possibility of having such deformations using PUPs is an interesting future research.

In conclusion, PUPs have the potential to be used as basis function of a FFD and generalized to GFFD. However, as the effect of a combination of basis functions is sometimes hard to determine, GFFD should be implemented in a user friendly environment and some already known good set-up should be found to be provided for the users. GFFD can be easily used for 3D dimensional meshes with arbitrary connectivity and no parameterization is needed although we have not provided these results in this paper.

References

[1] Richard Bartels, Ali Mahdavi-Amiri, Faramarz Samavati, and Nezam Mahdavi-Amiri. Diagrammatic approach for constructing multiresolution of primal subdivisions. *Computer Aided Geometric Design*, 51:4 – 29, 2017.

[2] Richard H Bartels, John C Beatty, and Brian A Barsky. *An introduction to splines for use in computer graphics and geometric modeling.* Morgan Kaufmann, 1995.

[3] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *IEEE transactions on visualization and computer graphics*, 14(1):213–230, 2008.

[4] Jack Caron and David Mould. Texture synthesis using label assignment over a graph. *Computers & Graphics*, 39:24–36, 2014.

[5] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 31–36, New York, NY, USA, 2001. ACM.

[6] Michael S Floater. Mean value coordinates. *Computer aided geometric design*, 20(1):19–27, 2003.

[7] Sarah F. F. Gibson and Brian Mirtich. A survey of deformable modeling in computer graphics. Technical report, Mitsubishi Electric Research Laboratories, 1997.

[8] William M. Hsu, John F. Hughes, and Henry Kaufman. Direct manipulation of free-form deformations. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, pages 177–184, 1992.

[9] Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 30(4):78:1–78:8, July 2011.

[10] Doug L James and Christopher D Twigg. Skinning mesh animations. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 399–407. ACM, 2005.

[11] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic coordinates for character articulation. In *ACM Transactions on Graphics (TOG)*, volume 26, page 71. ACM, 2007.

[12] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 561–566. ACM, 2005.

[13] Yaron Lipman, David Levin, and Daniel Cohen-Or. Green coordinates. In *ACM Transactions on Graphics (TOG)*, volume 27, page 78. ACM, 2008.

[14] Ali Mahdavi-Amiri and Faramarz Samavati. Connectivity maps for subdivision surfaces. In *GRAPP/IVAPP*, pages 26–37, 2012.

[15] Ali Mahdavi-Amiri and Faramarz Famil Samavati. ACM: Atlas of connectivity maps for semiregular models. In *Proceedings of Graphics Interface 2013*, GI '13, pages 99–107, 2013.

[16] Ali Mahdavi-Amiri and Faramarz Famil Samavati. Atlas of connectivity maps. *Computers & Graphics*, 39(0):1 – 11, 2014.

[17] Amirhessam Moltaji, Adam Runions, and Faramarz Samavati. Subdivision and multiresolution for pups. *Computers & Graphics*, pages 1–14, 2017.

[18] Andrew Nealen, Olga Sorkine, Marc Alexa, and Daniel Cohen-Or. A sketch-based interface for detail-preserving mesh editing. In *ACM SIGGRAPH 2007 courses*, page 42. ACM, 2007.

[19] Luke J. Olsen and Faramarz F. Samavati. A discrete approach to multiresolution curves and surfaces. In *Proceedings of the 2008 International Conference on Computational Sciences and Its Applications*, pages 468–477. IEEE Computer Society, 2008.

[20] Luke J. Olsen, Faramarz F. Samavati, and Richard H. Bartels. Multiresolution for curves and surfaces based on constraining wavelets. *Computers & Graphics*, 31(3):449 – 462, 2007.

[21] R. Pusch and F. Samavati. Local constraint-based general surface deformation. In *Shape Modeling International Conference (SMI) 2010*, pages 256–260. IEEE Computer Society, Jun 21-23 2010.

[22] Adam Runions and Faramarz Samavati. *CINPACT-splines: A Class of C-Ininfty Curves with Compact Support*, pages 384–398. Cham, 2015.

[23] Adam Runions and Faramarz F. Samavati. Partition of unity parametrics: a framework for meta-modeling. *The Visual Computer*, 27(6-8):495–505, Jun 2011.

[24] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 151–160, New York, NY, USA, 1986. ACM.

[25] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 175–184, New York, NY, USA, 2004. ACM.

[26] Johannes Zimmermann, Andrew Nealen, and Marc Alexa. Silsketch: Automated sketch-based editing of surface meshes. In *Proceedings of the 4th Eurographics Workshop on Sketch-based Interfaces and Modeling*, SBIM '07, pages 23–30, New York, NY, USA, 2007. ACM.

[27] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *Proc. of SIGGRAPH '97*, pages 259–268, New York, NY, USA, 1997. ACM Press.